Shop Digital Ads Node JS Backend Documentation

Shops Digital Ads

Shops Digital Ads is a digital advertisement platform that connects users who own display screens with those who want to promote ads. The platform allows users to register their displays, upload ads, and manage ad campaigns. Ads approved by the admin are automatically downloaded to Android TV devices, where they are played until their expiry.

🧩 Project Structure

This project consists of the following components:

1. Backend

- **Technology**: Node.js with Express.js
- **Database**: MySQL
- **Functionality**:
    - User authentication and management
    - Display registration
    - Ad submission, approval, and scheduling
    - Income tracking for displays
    - Admin panel for approvals and user management
    - REST APIs for Flutter apps
- **GitHub**: Backend Repository

2. Mobile App (Flutter)

- **Platform**: Android & iOS
- **Features**:
    - Unified user system with both display registration and ad upload features
    - Register and manage display devices
    - Upload and manage ad campaigns

    - View earnings generated from ads running on personal displays
    - Track ad status and expiry
- **GitHub**: Mobile App Repository

3. Android TV App (Flutter)

- **Platform**: Android TV
- **Features**:
    - Syncs and downloads approved ads for offline playback
    - Plays ads in a loop until their expiry
    - Designed for full-screen immersive ad display
- **GitHub**: TV App Repository

👥 User Functionality

All users on the platform can:

- Register display screens to show ads
- Upload advertisements to be shown on other users' displays
- Earn income when ads are shown on their registered displays
- Track the income generated from ad views on their screens

There are no strict roles like "Display Owner" or "Ads Uploader". Every user has access to all functionalities.

🔁 Workflow

1. User signs up via the **Mobile App**
2. User can:
    a. Register a **Display** to host ads
    b. **Upload Ads** to be played on available displays
3. Admin reviews and approves submitted ads
4. Approved ads are downloaded to the **Android TV App**
5. Ads play in a loop until the defined **expiry date**
6. Users earn based on the ads displayed on their registered screens

🔧 Technologies Used

Layer	Technology
Backend	Node.js (Express)
Database	MySQL
Mobile App	Flutter
TV App	Flutter
Admin Panel	Web Interface (Upcoming)

📦 Setup Instructions

Backend

git clone : *https://github.com/shopsdigitalads/ShopDigiNodeBackend*
cd ShopsDigitalAds-Backend
npm install
Configure your .env file
npm test

Flutter Mobile App

git clone : *https://github.com/shopsdigitalads/ShopDigiApp*
cd ShopsDigitalAds-Mobile
flutter pub get
flutter run

Flutter Android TV App

git clone : *https://github.com/shopsdigitalads/SDA-VIDEO-APP*
cd ShopsDigitalAds-TV
flutter pub get
flutter run

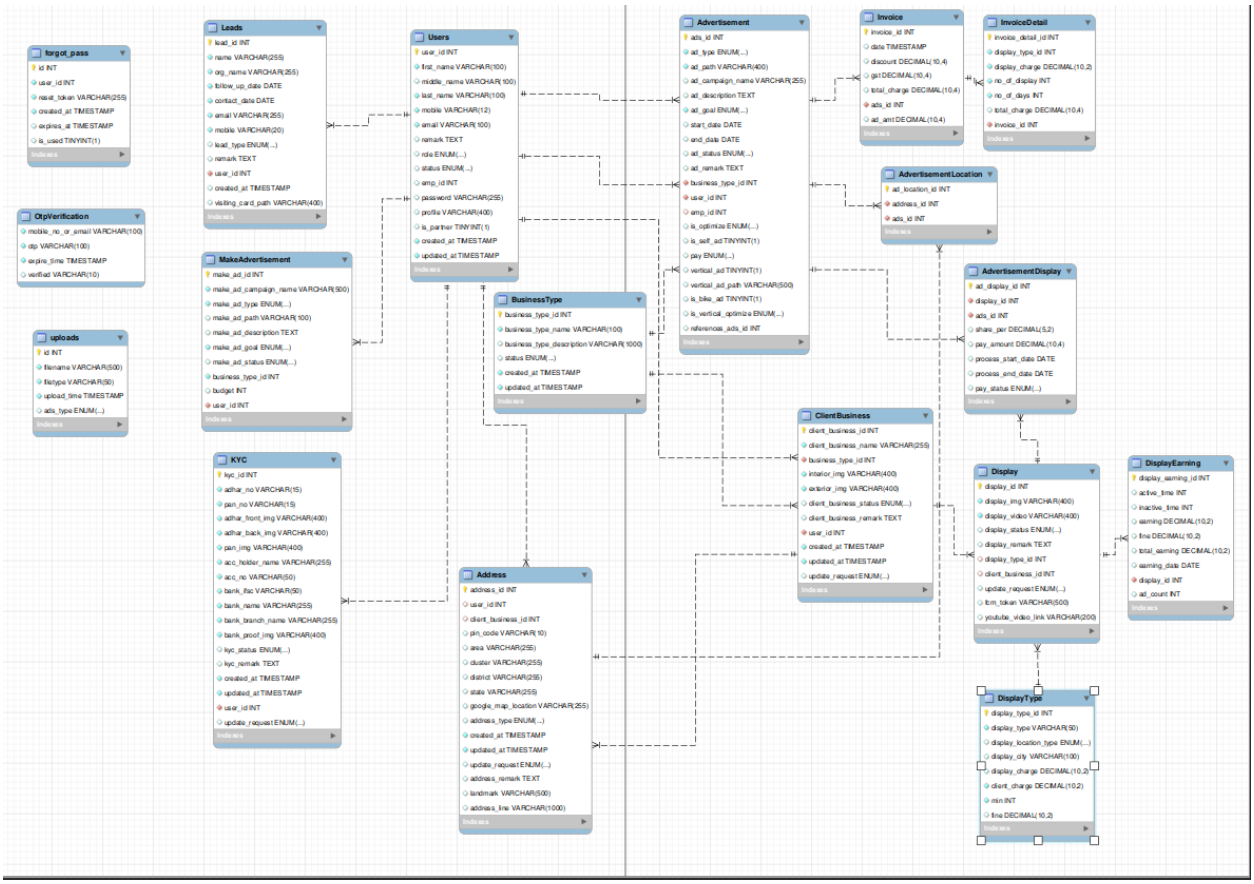
📧 Contact

Shop Digital Ads

Developer | Shops Digital Ads

📩 Email: [Your Email Here]

Database Schema :



Project Structure

```
project-root/
│
├── Handlers/              # All controller logic organized by modules
│   ├── Authentication/    # Handles user authentication (login, JWT, etc.)
│   ├── Client/            # Client-related business logic
│   │   └── [ModuleName]/  # Separate folders per client feature/module
│   ├── Display/           # Handlers for Display App functionality
│   ├── Employee/          # Handles employee-related operations
│   ├── Notification/      # Logic for sending notifications to apps
│   └── Video/             # Handlers for video compression and related tasks
│
├── Routers/               # Route definitions mapped to handlers
│   ├── Authentication/    # Routes for login, registration, etc.
│   ├── Client/            # Client-related API endpoints
│   │   └── [ModuleName]/  # Separate routers for client modules
│   ├── Display/           # API endpoints for Display App
│   ├── Employee/          # API endpoints for employee actions
│   ├── Notification/      # Routes to trigger notifications
│   └── Video/             # Routes for video compression/upload/etc.
│
├── Utils/                 # Utility/helper functions used across modules
│   └── [filename].js      # (e.g., db.js, responseFormatter.js, etc.)
│
├── .env                   # Environment variables
```

```
├── package.json          # Project dependencies and scripts
└── index.js              # Entry point of the application
```

Folder Descriptions

- **Handlers/**: This folder contains all the controller logic. Each module is separated into its own folder for better maintainability.
    - **Authentication/**: Manages login, registration, and JWT handling.
    - **Client/**: Contains submodules and handlers for client-related features.
    - **Display/**: Manages the Display App's logic (e.g., fetching/displaying ads).
    - **Employee/**: Handles employee creation, management, etc.
    - **Notification/**: Handles push notifications sent to the mobile or TV apps.
    - **Video/**: Contains logic related to video compression and management.
- **Routers/**: Contains Express route definitions that map HTTP endpoints to their respective handlers.
    - Mirrors the structure of the Handlers/ folder for consistency.
- **Utils/**: Utility functions that are reused throughout the project. This may include:
    - Database configuration and connection
    - Response formatting helpers
    - Middleware utilities (e.g., auth middleware)
- **server.js**: Main entry point of the application. Sets up the Express app, applies middlewares, connects to the database, and mounts routers.

✅ Standard Format for Express Handlers

```
/**
 * @desc       [Short description of what this handler does]
 * @route      [HTTP_METHOD] /api/endpoint
 * @access     [Public/Private]
 */
static functionName = async (req, res) => {
   try {
      // 1. Input Validation
      // Validate and destructure required inputs from req.body, req.params, req.files,
etc.

      // 2. Business Logic
      // - Handle files (upload, rename, move, etc.)
      // - Interact with MySQL using `pool.query()`
      // - Call any helper functions or other services
      // - Handle nested logic if needed

      // 3. Return Success Response
      return res.status(200).json({
         status: true,
         message: "Success message",
         data: { ... }, // Optional data
      });

   } catch (error) {
      // 4. Error Handling
      console.error(error);
      return res.status(500).json({
         status: false,
         message: "An internal error occurred",
         details: error.message, // Optional for debugging
      });
   }
};
```

🧱 Explanation with Your Examples

Let’s break down what each part usually does using examples:

◇ Input Validation Pattern

```
const { field1, field2 } = req.body;
if (!field1 || !field2) {
   return res.status(400).json({
      status: false,
      message: "Data Missing"
   });
}
```

- Ensures required inputs are present before any processing begins.
- Also includes file validation like req.files['fieldName'].

◇ File Handling Pattern

```
const old_path = req.files['field'][0].path;
const new_path = path.join(destination, 'filename.jpg');
fs.renameSync(old_path, new_path);
```

- Creates folder (if it doesn’t exist)
- Renames and stores files in structured locations
- Stores file path relative to base

◇ Database Interaction Pattern

```
const [result] = await pool.query('SQL_QUERY', [params]);
```

- Uses await pool.query() for queries
- Validates affectedRows or insertId for success
- Optionally calls another handler (e.g. Address.add())

◇ Success Response

```
return res.status(201).json({
   status: true,
   message: "Operation Successful",
   data: { ... }
});
```

◇ Error Handling

```
catch (error) {
   console.error(error);
   return res.status(500).json({
      status: false,
      message: "An internal error occurred",
      details: error.message
   });
}
```

- Catches all errors and logs them
- Sends consistent 500 response with optional debug info

Suggested Commented Template

You can use this as a copy-paste boilerplate for new handlers:

```
/**
 * @desc       Add a new business
 * @route      POST /api/client/business
 * @access     Private
 */
static addBusiness = async (req, res) => {
   try {
      // Input Validation
      const { field1, field2 } = req.body;
      if (!field1 || !field2 || !req.files['someFile']) {
         return res.status(400).json({ status: false, message: "Data Missing" });
      }

      // Folder creation
      const folder = path.resolve(__dirname, `../some/path`);
      if (!fs.existsSync(folder)) fs.mkdirSync(folder, { recursive: true });

      // File rename/move
      const filePath = path.join(folder, 'file.jpg');
      fs.renameSync(req.files['someFile'][0].path, filePath);

      // Insert into DB
      const query = `INSERT INTO table (...) VALUES (...)`;
      const [result] = await pool.query(query, [values]);

      // Return success
      return res.status(201).json({
         status: true,
         message: "Successfully Added"
      });

   } catch (error) {
      console.error(error);
      return res.status(500).json({
         status: false,
```

```
        message: "Internal Server Error",
        details: error.message
      });
  }
};
```

API Documentation

Auth

POST SendOTP

Open Request

http://178.16.137.175:5000/auth

Body raw (json)

json

```
{
    "receive": "7709869270",
    "val": "Mobile"
}
```

PUT VerifyOTP

Open Request

http://localhost:5000/auth

Body raw (json)

json

```
{
    "receive":7777777777,
    "otp":734454
}
```

GET Display Login

Open Request

http://localhost:5000/auth/display/100009

PUT Display OTP Verify

Open Request

http://localhost:5000/auth/display

Body raw (json)

json

```
{
    "receive":"7",
    "otp":697327
}
```

Registration

POST Register

Open Request

http://localhost:5000/client

Authorization Bearer Token

Token

<token>

Body raw (json)

json

```
{
    "first_name":"Abhishek",
    "middle_name":"Purushottam",
    "last_name":"Bhoyar",
    "mobile":"1111111111",
    "email":"abhibhoyar141@gmail.com",
    "role":"Client"
}
```

PUT Update

Open Request

http://localhost:5000/client

Authorization Bearer Token

Token

<token>

Body raw (json)

json

```
{
    "update_field":["first_name","last_name"],
    "update_data":["Abhi","Bho",1]
}
```

Address

POST Add Address

Open Request

http://localhost:5000/address

Authorization Bearer Token

Token
<token>

Body raw (json)

json
```
{
    "pin_code":222222,
    "area":"Dagegaon Rangari",
    "cluster":"Saoner",
    "district":"Nagpr",
    "state":"Maharashtra",
    "address_type":"Home",
    "user_id":1
}
```

PUT Update Address

Open Request
http://localhost:5000/address

Authorization Bearer Token

Token
<token>

Body raw (json)

json
```
{
    "update_field":["pin_code","area"],
    "update_data":[11111,"yccce",1]
}
```

GET Business Address

Open Request

http://localhost:5000/address/ads/11

Authorization Bearer Token

Token
<token>

GET Get Address of user

Open Request
http://localhost:5000/address/31

Authorization Bearer Token

Token
<token>

KYC

POST Add KYC

Open Request
http://178.16.137.175:5000/kyc

Authorization Bearer Token

Token
<token>

Body form-data

adhar_no
54
pan_no
asdl
acc_holder_name
lkjds
acc_no
slkd
bank_ifsc
skd
bank_branch_name
dshk
name
Abhi
user_id
1
adhar_front_img
Adhar_front_img.png
adhar_back_img
Adhar_front_back.png
pan_img
Pan_img.png
bank_proof_img
Bank_proof_img.png
bank_name
ksjdf

PUT Update KYC

Open Request
http://localhost:5000/k

Authorization Bearer Token

Token

<token>

Body form-data

update_field
["adhar_no","pan_no"]
update_data
["asd","sdlf"]
name
Abhi
user_id
1
adhar_front_img
Adhar_front_im.png

GET Get KYC of user

Open Request
http://localhost:5000/kyc/31

Authorization Bearer Token

Token
<token>

Business

POST Add Business

Open Request
http://localhost:5000/business

Authorization Bearer Token

Token
<token>

Body form-data

client_business_name
sadf
business_type_id
1
address_type
Business
name
Abhi
user_id
1
interior_img
Interior_img.png
exterior_img
Exterior_img.png
pin_code
110001
area
Bengali Market
cluster
New Delhi
district
Central Delhi
state
Delhi

GET Business Types

Open Request
http://localhost:5000/business/types

Authorization Bearer Token

Token
<token>

GET Get User Business with Display

Open Request
http://localhost:5000/business/1

Authorization Bearer Token

Token
<token>

GET Fetch Business update request

Open Request
http://localhost:5000/business/update_request/1

Authorization Bearer Token

Token
<token>

Display

POST Add Display

Open Request

Authorization Bearer Token

Token
<token>

GET Display Types

Open Request
http://localhost:5000/display/types

Authorization Bearer Token

Token
<token>

Body raw (json)

json
```
{
    "address_ids":[20],
    "business_type_id":1
}
```

POST Display Using addrss

Open Request
http://localhost:5000/display/ads

Authorization Bearer Token

Token
<token>

Body raw (json)

```
json
{
    "address_ids":[33,36]
}
```

POST Fetch Ads for Display

Open Request

http://localhost:5000/display/app/ads

Authorization Bearer Token

Token

<token>

Body raw (json)

```
json
{
    "display_id":100001,
    "ads_ids":[]
}
```

GET Display History

Open Request

http://localhost:5000/display/history/1

Authorization Bearer Token

Token

<token>

POST download ads

Open Request

http://localhost:5000/display/app/ads/download

Authorization Bearer Token

Token

<token>

Body raw (json)

View More

json


```
{
    "display_id":100001,
    "ads": [
        {
            "ads_id": 100013,
            "ad_type": "IMAGE",
            "ad_path":
"Media/Client/user_id_first_name_middle_name_last_name/Advertisement/ad_file_1738661218044.jpg
",
            "start_date": "2025-01-31T18:30:00.000Z",
            "end_date": "2025-02-19T18:30:00.000Z"
        },
        {
            "ads_id": 100014,
            "ad_type": "VIDEO",
            "ad_path":
"Media/Client/user_id_first_name_middle_name_last_name/Advertisement/ad_file_1738661218044.jpg
",
            "start_date": "2025-02-03T18:30:00.000Z",
            "end_date": "2025-02-27T18:30:00.000Z"
        },
        {
            "ads_id": 100015,
            "ad_type": "IMAGE",
            "ad_path":
"Media/Client/user_id_first_name_middle_name_last_name/Advertisement/ad_file_1738661218044.jpg
",
            "start_date": "2025-02-03T18:30:00.000Z",
            "end_date": "2025-03-20T18:30:00.000Z"
```

```
        },
        {
            "ads_id": 100016,
            "ad_type": "IMAGE",
            "ad_path":
"Media/Client/user_id_first_name_middle_name_last_name/Advertisement/ad_file_1738661218044.jpg
",
            "start_date": "2025-02-06T18:30:00.000Z",
            "end_date": "2025-02-27T18:30:00.000Z"
        },
        {
            "ads_id": 100018,
            "ad_type": "VIDEO",
            "ad_path":
"Media/Client/user_id_first_name_middle_name_last_name/Advertisement/ad_file_1738661218044.jpg
",

            "start_date": "2025-02-07T18:30:00.000Z",
            "end_date": "2025-02-17T18:30:00.000Z"
        }
    ]
}
```


POST Display Earning

Open Request

http://localhost:5000/display/earning

Authorization Bearer Token

Token

<token>

Body raw (json)

View More

json


```
  {
  "display_status":[
  {
    "display_status_id": 3,
```

```
    "date": "2025-02-08",
    "start_time": "17:32:36",
    "end_time": "17:59:36",
    "status": "Not Sent"
  },
  {
    "display_status_id": 4,
    "date": "2025-02-08",
    "start_time": "17:33:36",
    "end_time": "17:34:37",
    "status": "Not Sent"
  }
],
"ad_count":2,
"display_id":100002
}
```


PUT update request

Open Request

http://localhost:5000/display/update_request

Authorization Bearer Token

Token

<token>

Body raw (json)

json


```
{
    "remark":"Testing",
    "display_id":100001
}
```


GET Fetch Display Update Request

Open Request

http://localhost:5000/display/update_request/1

Authorization Bearer Token

Token
<token>

Ads

POST Create Add

Open Request
http://localhost:5000/ads/create

Authorization Bearer Token

Token
<token>

Body raw (json)

json
```
{
    "make_ad_type":"IMAGE",
    "make_ad_description":"adsf",
    "make_ad_goal":"Brand awareness",
    "business_type_id":1,
    "budget":"25",
    "user_id":1
}
```

POST Upload Ad

Open Request
http://localhost:5000/ads/upload

Authorization Bearer Token

Token
<token>

Body form-data

ad_type
IMAGE
ad_description
SFD
ad_goal
Brand awareness
start_date
2025-12-12
end_date
2025-12-12
business_type_id
1
user_id
1
name
Abhi
ad_file
Ad_file..png

POST Add Ads Display

Open Request
http://localhost:5000/ads/display

Authorization Bearer Token

Token

<token>

Body raw (json)

json

```
{
    "displays":[ 1, 2, 5 ],
    "ad_id":37
}
```

GET Get Ads Of User

Open Request

http://localhost:5000/ads/31

Authorization Bearer Token

Token

<token>

GET Get Ad Details

Open Request

http://localhost:5000/ads/details/53

Authorization Bearer Token

Token

<token>

POST Ads Display

Open Request

http://localhost:5000/ads/ad_display

Authorization Bearer Token

Token
<token>

Body raw (json)

json
```
{
    "address_ids":[33,37],
    "ad_id":53
}
```

Employee

GET Fetch Client of Employee

Open Request
http://localhost:5000/employee/clients/2

Authorization Bearer Token

Token
<token>

Notification

POST Send Published Ad Notification

Open Request

http://178.16.137.175:5000/notification

Authorization Bearer Token

Token

<token>

Video

GET Video Otimize

Open Request

http://localhost:5000/video/100059

Authorization Bearer Token

Token

<token>