# Morat E-commerce Website

## Documentation

Important Note:

Any request properties with a **?** is optional depending on what type of function you are trying to make.

**Note:** please update env variables for email settings of nodemailer.

**Note:** please update env variables for cloudinary.

**For Example:** {property1 , property2?, property3?:String, property4?:Number = 2}

Property1 can’t be excluded in the request.

Property2 can be add or excluded according to your needs.

property3 has a type of String.

Property4 has a type of Number with a default value of 2.

### Code For Converting Image to base64:

function formInputImage(e) {

    const target = e.target;

    const file = target.files[0];

    const fileReader = new FileReader();

    fileReader.readAsDataURL(file);

    fileReader.onload = () => {

      const { name, type, size } = file;

      const image = { name, type, size, base64: fileReader.result };

// setProductForm is a useState

      setProductForm({

        ...productForm,

        image,

      });

    };

  }

### Banner:

**Create Banner:**

Details: Used to create a Banner

Url: /api/createBanner

Request: {

title? {

      text: String = "",

      color: String = "#242424",

    },

    subTitle?: {

      text: String = "",

      color: String = "#242424",

    },

    image: {

      size: String,

      name: String,

      type: String,

// convert the image to base64

base64: String,

    },

    btn?: {

      text: String = "",

      color: String = "#ffffff",

      bgColor: String = "#242424",

      link: String = "",

    },

}

Response: {

bannerData: Object,

msg: String,

}

**Update Banner:**

Details: Used to update a single banner.

Url: /api/updateBanner

Request: {

id: mongooseId,

title? {

      text: String = "",

      color: String = "#242424",

    },

    subTitle?: {

      text: String = "",

      color: String = "#242424",

    },

    image?: {

      size: String,

      name: String,

      type: String,

// convert the image to base64

base64: String,

    },

    btn?: {

      text: String = "",

      color: String = "#ffffff",

      bgColor: String = "#242424",

      link: String = "",

    },

}

Response: {

bannerData: Object,

msg: String,

}

**Delete Banner:**

Details: Used to delete a single banner.

Url: /api/deleteBanner

Request: { id: mongooseId }

Response: {

msg: String,

}

**Get All Banners:**

Details: Used to get all banners.

Url: /api/getAllBanners

Request: {}

Response: {

bannerData: Array,

msg: String,

}

**Get Banner:**

Details: Used to get a single banner.

Url: /api/getBanner

Request: { id: mongooseId }

Response: {

bannerData: Object,

msg: String,

}

### Category:

**Create Category:**

Details: Used to create a single category

Url: /api/createCategory

Request: { name: String }

Response: {

categoryData: Object,

msg: String,

}

**Update Category:**

Details:

Url: /api/updateCategory

Request: { id: mongooseId, name: String }

Response: {

categoryData: Object,

msg: String,

}

**Delete Category:**

Details:

Url: /api/deleteCategory

Request: { id: mongooseId }

Response: {

msg: String,

}

**Get All Categories:**

Details:

Url: /api/getAllCategories

Request: {}

Response: {

categoryData: Array,

msg: String,

}

**Get Category:**

Details:

Url: /api/getCategory

Request: { name: String }

Response: {

categoryData: Object,

msg: String,

}

### CategoryType:

**Create Category Type:**

Details: Used to create a category type that is linked to the category

Url: /api/createCategoryType

Request: {

    category: mongooseId,

    image?: {

      size: String,

      name: String,

      type: String,

// convert the image to base64

base64: String,

    },

    name: String,

    color?: String = "#242424",

  }

Response: {

categoryTypeData: Object,

msg: String

}

**Update Category Type:**

Details: Used to update category type that is linked to the category.

Url: /api/updateCategoryType

Request: {

    category: mongooseId,

    image?: {

      size: String,

      name: String,

      type: String,

// convert the image to base64

base64: String,

    },

    name: String,

    color?: String = "#242424",

  }

Response: {

categoryTypeData: Object,

msg: String

}

**Delete Category Type:**

Details: Used to delete a category type.

Url: /api/deleteCategoryType

Request: { id: String }

Response: {msg}

**Get All Category Types:**

Details: Used to get all category types that is linked to the category.

Url: /api/getAllCategoryTypes

Request: { category: mongo.id}

Response: {

categoryTypeData: Array,

msg: String

}

**Get Category Type:**

Details: Used to get a single category type by its name

Url: /api/getCategoryType

Request: { category: mongo.id, name: String }

Response: {

categoryTypeData: Object,

msg: String

}

### SubCategory:

**Create Sub Category:**

Details: Used to create a sub category linked to the category and category type.

Url: /api/createSubCategory

Request: {

    category: mongo.Id,

    categoryType: mongo.Id,

    name: String,

  }

Response: {

subCategoryData: Object,

msg: String

}

**Update Category Type:**

Details: Used to update a sub category using the sub category, category and category type id’s.

Url: /api/updateSubCategory

Request: {

id: mongo.Id,

    category: mongo.Id,

    categoryType: mongo.Id,

    name: String,

  }

Response: {

subCategoryData: Array,

msg: String

}

**Delete Sub Category:**

Details: Used to delete a sub category using its id.

Url: /api/deleteSubCategory

Request: { id: mongo.id }

Response: {msg: String}

**Get All Sub Categories:**

Details: Used to get all sub category by category or category type.

Url: /api/getAllSubCategories

Request: { category?: mongo.id, categoryType?: mongo.id }

Response: {

subCategoryData: Array,

msg: String

}

**Get Sub Category:**

Details: Used to get a sub category using the category or category type.

Url: /api/getSubCategory

Request: { category: mongo.id, categoryType: mongo.id, name: String }

Response: {

subCategoryData: Object,

msg: String

}

### Product:

**Create Product:**

Details: Used to create a product using its image, name, description, colors, category, category type, sub category and price. Discount featured and available quantity are optional.

Url: /api/createProduct

Request: {

    image: {

      size: String,

      name: String,

      type: String,

// convert the image to base64

base64: String,

    },

    name: String,

    description: String,

    colors: Array = [],

    category: mongoose.ObjectId,

    categoryType: mongoose.ObjectId,

    subCategory: mongoose.ObjectId,

    price: Number = 0,

    discount?: Number = 0,

    featured?: Boolean = false,

    availableQuantity?: Number = 0,

  }

Response: {productData: Object, msg: String}

**Update Product:**

Details: Used to update a product using its name, description, colors, category, category type, sub category and price. Image, discount, featured and available quantity are optional.

Url: /api/updateProduct

Request: {

id: mongo.id,

    Image?: {

      size: String,

      name: String,

      type: String,

// convert the image to base64

base64: String,

    },

    name: String,

    description: String,

    colors: Array = [],

    category: mongoose.ObjectId,

    categoryType: mongoose.ObjectId,

    subCategory: mongoose.ObjectId,

    price: Number = 0,

    discount?: Number = 0,

    featured?: Boolean = false,

    availableQuantity?: Number = 0,

    rating?: Number = 5,

  }

Response: {productData: Object, msg: String}

**Delete Product:**

Details: Used to delete a product using an id of the product.

Url: /api/deleteProduct

Request: { id: mongo.id }

Response: {productData: Object, msg: String}

**Get All Products:**

Details: Used to get all products by either name, category, category type, sub category, price with a minimum price or maximum price or featured. You can also limit the results displayed and also do pagination using totalpages and current page.

Url: /api/getAllProducts

Request: {

    name?: String = "",

    colors?: String,

    category?: String,

    categoryType?: String,

    subCategory?: String

    price? = { priceMin: Number = 0, priceMax: Number = 99999 },

    featured?: Boolean = false,

    currentPage?: Number = 1,

    limit?: Number = 6,

  }

Response: { products: Array, totalPages: Number, currentPage: Number }

**Get Product:**

Details: Get a single page using its id.

Url: /api/getProduct

Request: { id: mongo.id }

Response: {productData: Object, msg: String}

### Review:

**Create Review:**

Details: Used to create a review about the product it is linked to.

Url: /api/createReview

Request: {

    productID: mongoose.ObjectId,

    userID: mongoose.ObjectId,

    review: String,

    rating: Number,

  }

Response: {reviewData: Object, msg: String}

**Update Review:**

Details: Used to update a review about the product it is linked to

Url: /api/updateReview

Request: {

    productID: mongoose.ObjectId,

    userID: mongoose.ObjectId,

    review: String,

    rating: Number,

  }

Response: {reviewData, msg}

**Delete Review:**

Details: Used to delete a review.

Url: /api/deleteReview

Request: { id: mongo.id }

Response: {reviewData: Object, msg: String}

**Get All Reviews By Product:**

Details: Used to get all reviews about the product it is linked to by using the product id.

Url: /api/getAllReviewsByProduct

Request: { productID: mongo.id }

Response: {reviewData: Array, msg: String}

### User:

**Signup:**

Details: Used to signup a user using their name, email, password and role.

**Roles are user and admin.**

Url: /api/signup

Request: {

    name: String,

    email: String

    password: String,

    address?: String = "",

    phone?: String = "",

    role: type: String,

    refreshToken?: String = "",

  }

Response: {userData: Object, msg: String}

**Login:**

Details: Used to login a user using their email, password and role.

**Roles are user and admin.**

Url: /api/login

Request: {

    email: String

    password: String,

    role: type: String,

  }

Response: {userData: Object, msg: String}

**Update User:**

Details: Used to update a users details.

Url: /api/updateUser

Request: {

    name: String,

    email: String

newPassword: String,

confirmPassword: String,

    address?: String = "",

    phone?: String = "",

    refreshToken?: String = "",

  }

Response: {userData: Object, msg: String}

**Validate Token:**

Details: Used to validate if a user access token or refresh token is still valid.

**Note:** In the response a access token is given please update with this token in your cookies.

Url: /api/validateToken

Request: { accessToken: String }

Response: {

    tokenValidation: Boolean,

    validateAccessToken: Boolean,

    validateRefreshToken: Boolean,

    accessToken: String,

    msg: String,

  }

**Forgot Password:**

Details: Used to send a email to the users email account.

**Note:** please update env variables for email settings of nodemailer.

Url: /api/forgotPassword

Request: { email: String }

Response: {msg: String}

**Reset Password:**

Details: Used to update the password of the user by getting there id from reset token.

Url: /api/resetPassword

Request: { resetToken: String, newPassword: String, confirmPassword: String }

Response: {msg: String}

### Order:

**Create Order:**

Details: Used to create order.

Url: /api/createOrder

Request: {

items: Array,

total: Number,

shipping: Number,

    userID: mongoose.ObjectId,

    address: String,

    phone: String,

    status: String = "pending",

  }

Response: {orderData: object, msg: String}

**Update Order:**

Details: Used to update a order.

Url: /api/updateOrder

Request: {

id: mongo.id,

    address?: String,

    phone?: String,

    status?: String,

  }

Response: {orderData: object, msg: String}

**Get All Orders For User:**

Details: Used to get all orders by user id.

Url: /api/getAllOrdersForUser

Request: { userID: mongo.id }

Response: {orderData: object, msg: String}

**Get Order Details:**

Details: Used to get a order.

Url: /api/getOrderDetails

Request: { id: mongo.id }

Response: {orderData: object, msg: String}