

Specialist of Enterprise Level Apps, Large Scale Web Portals,

and iOS Apps & Android App Development

**Corporate Car Rental System**

Web Application & Mobile App Development Project

API Information Document

**iAssure International Technologies Pvt Ltd**

#323, Amanora Chambers, Hadapsar,

Pune, Maharashtra, India – 411013

www.iassureit.com | Contact: +91 – 9923 393 733

Founder & Managing Director: **Mr. Ashish Naik**

**Table of Contents**

[1. Company Settings](#_heading=h.gjdgxs)

[2. Users](#_heading=h.30j0zll)

3. [Notification](#_heading=h.1gf8i83) Template

4[. Master Data –](#_heading=h.40ew0vw)

4.[1. Corporate master](#_heading=h.2fk6b3p)

4.2 Supplier Master

4.3 Vendor Master

4.4 Employee Master

4.5 Driver Master

4.6 Guest Master

4.7 Category Master

4.8 Brand Master

4.9 Fuel Type Master

4.10 Model Maste  
4.11 Department Master

4.12 Designation Master

4.13 Package Type Master

4.14 Location Type Master

4.15 Module Master

4.16 Facility Master

4.17 Vehicle Master

4.18 Package Master

5. Access Management

6. Contract Management

# 1. Company Settings

**Description**

This is the collection / table for Company Information to be used all over the App.

|  |  |
| --- | --- |
| **Company Settings** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| companyName | String |
| companyContactNumber | String |
| companyName | String |
| companyAltContactNumber | String |
| companyEmail | String |
| companyAltEmail | String |
| companywebsite | String |
| companyaddress | String |
| countryCode | String |
| country | String |
| stateCode | String |
| state | String |
| district | String |
| taluka | String |
| city | String |
| pincode | String |
| companyLogo | String |
| **companyLocationsInfo** | **Array** |
| locationType | String |
| contactNumber | String |
| blockName | String |
| area | String |
| landmark | String |
| countryCode | String |
| country | String |
| stateCode | String |
| state | String |
| district | String |
| taluka | String |
| city | String |
| pincode | String |
| **bankDetails** | **Array** |
| accHolderName | String |
| accNickName | String |
| bankName | String |
| accType | String |
| branchName | String |
| accNumber | String |
| ifscCode | String |
| **taxSettings** | **Array** |
| taxType | String |
| taxRating | String |
| effectiveFrom | String |
| createdAt | new Date() |
| **paymentInfo** | **Array** |
| cashOnDelivery | Boolean |
| paytm | Boolean |
| UPI | Boolean |
| bankTransfer | Boolean |

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert Basic Information | Completed |
| 2. Get Company Settings Details. | Completed |
| 3. Update Basic Information | Completed |
| 4. Add Locations | Completed |
| 5. Get Single Location Details. | Completed |
| 6. Update Single Location Details | Completed |
| 7. Delete Single Location | Completed |
| 8. Add bank details | Completed |
| 9. Update Bank details | Completed |
| 10. Add tax | Completed |
| 11. Update Single tax | Completed |
| 12. Add payment gateway info | Completed |

--------------------------------------------------------------------------------1. Insert Basic Information

--------------------------------------------------------------------------------

Method : POST

URL : /api/companysettings

Input :

{

companyName : String,

companyContactNumber : String,

companyAltContactNumber: String,

companyEmail : String,

companyAltEmail : String,

companywebsite : String,

companyaddress : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

taluka : String,

city : String,

pincode : String,

companyLogo : String,

companyLocationsInfo : [

{

locationType : String,

contactNumber : String,

blockName : String,

area : String,

landmark : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

taluka : String,

city : String,

pincode : String,

}

],

bankDetails : [

{

accHolderName : String,

accNickName : String,

bankName : String,

accType : String,

branchName : String,

accNumber : String,

ifscCode : String,

}

],

taxSettings : [

{

taxType : String,

taxRating : String,

effectiveFrom : Date,

createdAt : Date,

}

],

paymentInfo : [

{

cashOnDelivery : Boolean,

paytm : Boolean,

UPI : Boolean,

bankTransfer : Boolean

}

]

}

Output :

if successful

{

message : "Company Added",

ID : String

}

else

{

message : "Something went wrong"

}

--------------------------------------------------------------------------------

2. Get Company Settings Details

--------------------------------------------------------------------------------

Method : GET

URL : /api/companysettings

------------------------------------------------------------------------------------------------------------------------

3. Update Basic Information

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/information

Input :

{

companyName : String,

companyContactNumber : String,

companyAltContactNumber: String,

companyEmail : String,

companyAltEmail : String,

companywebsite : String,

companyaddress : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

taluka : String,

city : String,

pincode : String,

companyLogo : String,

}

Output :

if successfully

{

message : "Company details updated"

}

else

{

message : "Company Setting Not Found"

}

--------------------------------------------------------------------------------

4. Add Locations

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/addLocation

Input :

{

companyId : ObjectID,

locationType : String,

contactNumber : String,

blockName : String,

area : String,

landmark : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

taluka : String,

city : String,

pincode : String,

}

Output : if successfully

{

message : "Company Locations Details added"

}

else

{

message : "Company Setting Not Found"

}

--------------------------------------------------------------------------------

5. Get Single Location Details

--------------------------------------------------------------------------------

Method : GET

URL : /api/companysettings/singleLocation/:locationID

Input : req.params.locationID

--------------------------------------------------------------------------------

6. Update Single Location Details

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/update\_location

Input : {

\_id : locationID,

locationType : String,

contactNumber : String,

blockName : String,

area : String,

landmark : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

taluka : String,

city : String,

pincode : String,

}

Output : if successfully

{

message : "Company Locations Details added"

}

else

{

message : "Company Locations Not found"

}

--------------------------------------------------------------------------------

7. Delete Single Location

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/companysettings/deleteLocation/:companyID/:locationID

--------------------------------------------------------------------------------

8. Add bank details

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/bankDetails

Input : {

\_id : companyId,

accHolderName : String,

accNickName : String,

bankName : String,

accType : String,

branchName : String,

accNumber : String,

ifscCode : String,

}

Output : if successfully

{

message : "Company bank details is added"

}

else

{

message : "Company Not found"

}

--------------------------------------------------------------------------------

9. Update Bank details

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/updateBankDetails

Input : {

\_id : companyId,

accHolderName : String,

accNickName : String,

bankName : String,

accType : String,

branchName : String,

accNumber : String,

ifscCode : String,

}

Output : if successfully

{

message : "Company bank details is updated"

}

else

{

message : "Company Not found"

}

--------------------------------------------------------------------------------

10. Add tax

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/taxSettings

Input : {

\_id : companyId,

taxType : String,

taxRating : String,

effectiveFrom : Date,

}

--------------------------------------------------------------------------------

11. Update single tax

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/updateTaxSettings

Input : {

\_id : companyId,

taxid : ObjectID,

taxType : String,

taxRating : String,

effectiveFrom : Date,

}

--------------------------------------------------------------------------------

12. Add payment gateway info

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/companysettings/addPaymentInfo

Input : { \_id : companyId,

paymentInfo : req.body.paymentInfo }

# 2. Users

List of APIs

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new user | Completed |
| 2. Details of user | Completed |
| 3. List details of all the users | Completed |
| 4. Update Role of the user | Completed |
| 5. Update User information (only profile Data) | Completed |
| 6. Delete particular user | Completed |
| 7. Delete all the users | Completed |
| 8. Login | Completed |
| 9. List users of specific company and specific role | Completed |

--------------------------------------------------------------------------------1. Insert new user (signup)

--------------------------------------------------------------------------------

Method : POST

URL : /api/users

Input :

{

"email" : String, // Mandatory Field

"pwd" : String, // Mandatory Field

"firstname" : String,

"lastname" : String, "mobNumber" : String,

"status" : String, // Either Active or Inactive

"otpMobile" : String,

"optEmail" : String,

"spoc" : Boolean,

"companyID" : Number,

"company\_ID" : String, // \_id of companysettings

"role" : Array, // Mandatory Field

}

Output :

If successfully

{

message : "User Created",

ID : 'String' // \_id of document

}

--------------------------------------------------------------------------------2. Details of user

--------------------------------------------------------------------------------

Method : GET

URL : /api/users/:userID

Input : - NA-

Output :

If successfully

{

"profile": {

"firstname" : String,

"lastname" : String,

"fullName" : String, "emailId" : String,

"mobNumber" : String,

"userCode" : String,

"status" : String,

"companyID" : Number,

"company\_ID" : String, //\_id of companysettings

},

"roles": [

String

],

"\_id": String

}

--------------------------------------------------------------------------------3. List details of all the users

--------------------------------------------------------------------------------

Method : GET

URL : /api/users/list

Input : - NA-

Output :

If successfully

[

{

"profile": {

"firstname" : String,

"lastname" : String,

"fullName" : String,

"emailId" : String,

"mobNumber" : String,

"userCode" : String,

"status" : String,

"spoc" : Boolean,

"companyID" : Number,

"company\_ID" : String, //\_id of companysettings

},

"roles": [

String

],

"\_id": String

}

]

--------------------------------------------------------------------------------4. Update Role of the user

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/users/:rolestatus

Description : rolestatus can be either 'assign' or 'remove'

Input : - NA-

Output :

If successfully assign

"Role Assigned"

If successfully remove

"Role Removed"

If not

"Something went wrong."

--------------------------------------------------------------------------------5. Update User information (only profile Data)

--------------------------------------------------------------------------------

Method : PUT

URL : /api/users

Input :

{

firstname : String,

lastname : String,

mobNumber : String,

status : String,

otpMobile : String,

optEmail : String,

spoc : Boolean,

companyID : Number,

company\_ID : String, //Either empty or \_id of companysettings

}

Output :

If successfully

"User Updated"

If not

"Something went wrong."

-------------------------------------------------------------------------------- 6. Delete particular user

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/users/delete/:userID

Note : Only Status = 'Inactive' can be deleted

Input : - NA -

Output :

If successfully

"User Deleted"

If not

"Something went wrong."

--------------------------------------------------------------------------------7. Delete all the users

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/users/delete

Input : - NA -

Output :

If successfully

"All Users Deleted"

If not

"Something went wrong."

--------------------------------------------------------------------------------8. Login

--------------------------------------------------------------------------------

Method : POST

URL : /api/users/details

Input :

{

email : String,

pwd : String

}

Output :

If successfully

{

token : token,

ID : String

}

If not

"Something went wrong."

--------------------------------------------------------------------------------9. List users of specific company and specific role

--------------------------------------------------------------------------------

Method : GET

URL : /api/list/:company\_ID/:role

Input : - NA -

Output :

[

{

"profile": {

"firstname" : String,

"lastname" : String,

"fullName" : String,

"emailId" : String,

"mobNumber" : String,

"userCode" : String,

"status" : String,

"spoc" : Boolean,

"companyID" : Number,

"company\_ID" : String, //\_id of companysettings

},

"roles": {

[String],

}

}

]

# 3. Notification Template

List of APIs

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new masternotification | Completed |
| 2. Details of masternotification | Completed |
| 3. List details of all the masternotifications | Completed |
| 4. Update masternotification details | Completed |
| 5. Delete particular masternotification | Completed |
| 6. Delete all the masternotifications | Completed |

-----------------------------------------------------------------------------------------------------

1. Insert new masternotification

-----------------------------------------------------------------------------------------------------

Method : POST

URL : /api/masternotifications

Note : templateName and templateType combination should be unique

Input :

{

templateType : String,

templateName : String,

subject : String,

content : String,

createdBy : String, // Either empty or \_id of companysettings

}

Output :

If successfully

{

message : "Notification Details Added",

ID : String

}

-----------------------------------------------------------------------------------------------------

2. Details of masternotification

-----------------------------------------------------------------------------------------------------

Method : GET

URL : /api/masternotifications/:notificationmaster\_ID

Input : - NA-

Output :

If successfully

{

templateType : String,

templateName : String,

subject : String,

content : String,

createdAt : Date,

createdBy : String,

\_id : String

}

-----------------------------------------------------------------------------------------------------

3. List details of all the masternotifications

-----------------------------------------------------------------------------------------------------

Method : GET

URL : /api/masternotifications/list

Input : - NA-

Output :

If successfully

[{

templateType : String,

templateName : String,

subject : String,

content : String,

createdAt : Date,

createdBy : String,

\_id : String

}]

-----------------------------------------------------------------------------------------------------

4. Update masternotification details

-----------------------------------------------------------------------------------------------------

Method : PUT

URL : /api/masternotifications/

Input :

{

id : String,

templateType : String,

templateName : String,

subject : String,

content : String

}

Output :

If successfully

"RoMaster notifications Updated"

If not

"Master notifications Found"

-----------------------------------------------------------------------------------------------------

5. Delete particular masternotification

-----------------------------------------------------------------------------------------------------

Method : DELETE

URL : /api/masternotifications/:notificationmaster\_ID

Input : - NA -

Output :

If successfully

"Master notification deleted"

-----------------------------------------------------------------------------------------------------

6. Delete all the masternotification

-----------------------------------------------------------------------------------------------------

Method : DELETE

URL : /api/masternotifications

Input : - NA -

Output :

If successfully

**"All Master notification deleted"**

# 4. Master Data

## 4.1 Corporate Master

|  |  |
| --- | --- |
| **Corporate Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| corporateCode | AutoIncrement 1, 2, 3 |
| companyName | String |
| groupName | String |
| CIN | String |
| COI | String |
| TAN | String |
| companyLogo | String |
| website | String |
| companyPhone | String |
| companyEmail | String |
| **locations** | **Array** |
| locationType | String |
| branchCode | Number |
| addressLine1 | String |
| addressLine2 | String |
| countryCode | String |
| country | String |
| stateCode | String |
| state | String |
| district |  |
| city | String |
| area | String |
| pincode | Number |
| GSTIN | String |
| GSTDocument | Array |
| PAN | String |
| PANDocument | Array |
| **contactPerson** | **Array** |
| branchCode | Number |
| Name | String |
| Phone | String |
| Alternate Phone | String |
| Email | String |
| Department | String |
| Designation | String |
| EmployeeID | String |
| userID | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | new Date() |
| createdBy | String |
| **updateLog** | **Array** |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new Corporate | Completed |
| 2. List of all corporates | Completed |
| 3. Count of all corporates | Completed |
| 4. Filter corporates | Completed |
| 5. Get details of single corporate | Completed |
| 6. Update single corporate | Completed |
| 7. Add corporate location | Completed |
| 8. Get details of single location | Completed |
| 9. Update single location | Completed |
| 10. Add corporate contact person | Completed |
| 11. Get details of single contact person | Completed |
| 12. Update single contact person | Completed |
| 13. Delete single corporate | Completed |
| 14. Delete single location for corporate | Completed |
| 15. Delete single contact person for corporate | Completed |

--------------------------------------------------------------------------------

1. Insert new **Corporate**

--------------------------------------------------------------------------------

Method : POST

URL : /api/corporate/post

Input :

{

\_id : new mongoose.Types.ObjectId(),

corporateCode : Auto increament number,

companyName : String,

groupName : String,

CIN : String,

COI : Array,

TAN : String,

companyLogo : Array,

website : String,

companyPhone : String,

companyEmail : String,

createdBy : {type:mongoose.Schema.Types.ObjectId,

ref: 'users' },

createdAt : new Date()

}

Output :

If successfully

{

created : true,

ID : data.\_id

}

If duplicate exists,

{ duplicated : true }

--------------------------------------------------------------------------------

2. List of all corporates

--------------------------------------------------------------------------------

Method : GET

URL : /api/corporate/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all corporates

--------------------------------------------------------------------------------

Method : GET

URL : /api/corporate/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter corporates

--------------------------------------------------------------------------------

Method : POST

URL : /api/corporate/get/filterEntities

Input : {

stateCode : “MH”,

district : “Pune”,

initial : “A”

}

--------------------------------------------------------------------------------

5. Get details of single corporate

--------------------------------------------------------------------------------

Method : GET

URL : /api/corporate/get/one/:corporateID

--------------------------------------------------------------------------------

6. Update single corporate

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/patch

Input : {

\_id : objectID,

companyName : String,

groupName : String,

CIN : String,

COI : Array,

TAN : String,

companyLogo : Array,

website : String,

companyPhone : String,

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId,

ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Add corporate location

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/patch/addLocation

Input : {

\_id : objectID,

locationType : String,

branchCode : Number,

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

GSTIN : String,

GSTDocument : Array,

PAN : String,

PANDocument : Array

}

Output :

If successfully

{ created : true }

If not

{ created : false }

--------------------------------------------------------------------------------

8. Get details of single location

--------------------------------------------------------------------------------

Method : POST

URL : /api/corporate/post/singleLocation

Input : {

entityID : objectID,

locationID : ObjectID

}

--------------------------------------------------------------------------------

9. Update single location

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/patch/updateSingleLocation

Inut : {

entityID : objectID,

locationID : ObjectID,

locationType : String,

branchCode : Number,

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

GSTIN : String,

GSTDocument : Array,

PAN : String,

PANDocument : Array

}

Output :

If successfully

{ updated : true }

If not

{ updated : false }

--------------------------------------------------------------------------------

10. Add corporate contact person

-----------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/patch/addContact

Inut : {

entityID : objectID,

branchCode : Number,

name : String,

phone : String,

altPhone : String,

email : String,

department : String,

designation : String,

employeeID : String,

userID : { type: mongoose.Schema.Types.ObjectId,

ref: 'users' }

}

Output :

If successfully

{

created : true,

}

If not

{ created : false }

--------------------------------------------------------------------------------

11. Get details of single contact person

--------------------------------------------------------------------------------

Method : POST

URL : /api/corporate/post/singleContact

Input : {

entityID : objectID,

contactID : ObjectID

}

--------------------------------------------------------------------------------

12. Update single contact person

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/patch/updateSingleContact

Input : {

entityID : objectID,

contactID : ObjectID,

branchCode : Number,

name : String,

phone : String,

altPhone : String,

email : String,

department : String,

designation : String,

employeeID : String

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

13. Delete single corporate

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/corporate/delete/:corporateID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

--------------------------------------------------------------------------------

14. Delete single location for corporate

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/delete/:corporateID/:locationID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

--------------------------------------------------------------------------------

15. Delete single contact person for corporate

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/corporate/delete/:corporateID/:contactID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

## 4.2 Supplier Master

|  |  |
| --- | --- |
| **Supplier Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| supplierCode | AutoIncrement 1, 2, 3 |
| supplierOf | Vendor id |
| companyName | String |
| groupName | String |
| CIN | String |
| COI | String |
| TAN | String |
| companyLogo | String |
| website | String |
| companyPhone | String |
| companyEmail | String |
| **locations** | **Array** |
| locationType | String |
| branchCode | Number |
| addressLine1 | String |
| addressLine2 | String |
| countryCode | String |
| country | String |
| stateCode | String |
| state | String |
| district |  |
| city | String |
| area | String |
| pincode | Number |
| GSTIN | String |
| GSTDocument | Array |
| PAN | String |
| PANDocument | Array |
| **contactPerson** | **Array** |
| branchCode | Number |
| Name | String |
| Phone | String |
| Alternate Phone | String |
| Email | String |
| Department | String |
| Designation | String |
| EmployeeID | String |
| userID | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | new Date() |
| createdBy | String |
| **updateLog** | **Array** |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new suppliers | Completed |
| 2. List of all suppliers | Completed |
| 3. Count of all suppliers | Completed |
| 4. Filter suppliers | Completed |
| 5. Get details of single supplier | Completed |
| 6. Update single supplier | Completed |
| 7. Add supplier location | Completed |
| 8. Get details of single location | Completed |
| 9. Update single location | Completed |
| 10. Add supplier contact person | Completed |
| 11. Get details of single contact person | Completed |
| 12. Update single contact person | Completed |
| 13. Delete single supplier | Completed |
| 14. Delete single location for supplier | Completed |
| 15. Delete single contact person for supplier | Completed |

--------------------------------------------------------------------------------

1. Insert new supplier

--------------------------------------------------------------------------------

Method : POST

URL : /api/supplier/post

Input :

{

\_id : new mongoose.Types.ObjectId(),

companyName : String,

groupName : String,

CIN : String,

COI : Array,

TAN : String,

companyLogo : Array,

website : String,

companyPhone : String,

companyEmail : String,

createdBy : { type: mongoose.Schema.Types.ObjectId,

ref: 'users' },

createdAt : new Date()

}

Output :

If successfully

{

created : true,

ID : data.\_id

}

If duplicate exists,

{ duplicated : true }

--------------------------------------------------------------------------------

2. List of all suppliers

--------------------------------------------------------------------------------

Method : GET

URL : /api/supplier/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all suppliers

--------------------------------------------------------------------------------

Method : GET

URL : /api/supplier/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter suppliers

--------------------------------------------------------------------------------

Method : POST

URL : /api/supplier/get/filterEntities

Input : {

stateCode : “MH”,

district : “Pune”,

initial : “A”

}

--------------------------------------------------------------------------------

5. Get details of single supplier

--------------------------------------------------------------------------------

Method : GET

URL : /api/supplier/get/one/:supplierID

--------------------------------------------------------------------------------

6. Update single supplier

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/patch

Input : {

\_id : objectID,

companyName : String,

groupName : String,

CIN : String,

COI : Array,

TAN : String,

companyLogo : Array,

website : String,

companyPhone : String,

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId,

ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Add supplier location

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/patch/addLocation

Input : {

\_id : objectID,

locationType : String,

branchCode : Number,

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

GSTIN : String,

GSTDocument : Array,

PAN : String,

PANDocument : Array

}

Output :

If successfully

{

created : true,

}

If not

{ created : false }

--------------------------------------------------------------------------------

8. Get details of single location

--------------------------------------------------------------------------------

Method : POST

URL : /api/supplier/post/singleLocation

Input : {

entityID : objectID,

locationID : ObjectID

}

--------------------------------------------------------------------------------

9. Update single location

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/patch/updateSingleLocation

Inut : {

entityID : objectID,

locationID : ObjectID,

locationType : String,

branchCode : Number,

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

GSTIN : String,

GSTDocument : Array,

PAN : String,

PANDocument : Array

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

10. Add supplier contact person

-----------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/patch/addContact

Inut : {

entityID : objectID,

branchCode : Number,

name : String,

phone : String,

altPhone : String,

email : String,

department : String,

designation : String,

employeeID : String

}

Output :

If successfully

{

created : true,

}

If not

{ created : false }

--------------------------------------------------------------------------------

11. Get details of single contact person

--------------------------------------------------------------------------------

Method : POST

URL : /api/supplier/post/singleContact

Input : {

entityID : objectID,

contactID : ObjectID

}

--------------------------------------------------------------------------------

12. Update single contact person

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/patch/updateSingleContact

Inut : {

entityID : objectID,

contactID : ObjectID,

branchCode : Number,

name : String,

phone : String,

altPhone : String,

email : String,

department : String,

designation : String,

employeeID : String

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

13. Delete single supplier

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/supplier/delete/:supplierID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

--------------------------------------------------------------------------------

14. Delete single location for supplier

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/delete/:supplierID/:locationID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

--------------------------------------------------------------------------------

15. Delete single contact person for supplier

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/supplier/delete/:supplierID/:contactID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

## 4.3 Vendor Master

|  |  |
| --- | --- |
| **Vendor Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| vendorCode | AutoIncrement 1, 2, 3 |
| companyName | String |
| groupName | String |
| CIN | String |
| COI | String |
| TAN | String |
| companyLogo | String |
| website | String |
| companyPhone | String |
| companyEmail | String |
| **locations** | **Array** |
| locationType | String |
| branchCode | Number |
| addressLine1 | String |
| addressLine2 | String |
| countryCode | String |
| country | String |
| stateCode | String |
| state | String |
| district |  |
| city | String |
| area | String |
| pincode | Number |
| GSTIN | String |
| GSTDocument | Array |
| PAN | String |
| PANDocument | Array |
| **contactPerson** | **Array** |
| branchCode | Number |
| Name | String |
| Phone | String |
| Alternate Phone | String |
| Email | String |
| Department | String |
| Designation | String |
| EmployeeID | String |
| userID | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | new Date() |
| createdBy | String |
| **updateLog** | **Array** |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new vendors | Completed |
| 2. List of all vendors | Completed |
| 3. Count of all vendors | Completed |
| 4. Filter vendors | Completed |
| 5. Get details of single vendor | Completed |
| 6. Update single vendor | Completed |
| 7. Add supplier location | Completed |
| 8. Get details of single location | Completed |
| 9. Update single location | Completed |
| 10. Add supplier contact person | Completed |
| 11. Get details of single contact person | Completed |
| 12. Update single contact person | Completed |
| 13. Delete single vendor | Completed |
| 14. Delete single location for vendor | Completed |
| 15. Delete single contact person for vendor | Completed |

--------------------------------------------------------------------------------

1. Insert new vendor

--------------------------------------------------------------------------------

Method : POST

URL : /api/vendor/post

Input :

{

\_id : new mongoose.Types.ObjectId(),

companyName : String,

groupName : String,

CIN : String,

COI : Array,

TAN : String,

companyLogo : Array,

website : String,

companyPhone : String,

companyEmail : String,

createdBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

createdAt : new Date()

}

Output :

If successfully

{

created : true,

ID : data.\_id

}

If duplicate exists,

{ duplicated : true }

--------------------------------------------------------------------------------

2. List of all vendors

--------------------------------------------------------------------------------

Method : GET

URL : /api/vendor/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all vendors

--------------------------------------------------------------------------------

Method : GET

URL : /api/vendor/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter vendors

--------------------------------------------------------------------------------

Method : POST

URL : /api/vendor/get/filterEntities

Input : {

stateCode : “MH”,

district : “Pune”,

initial : “A”

}

--------------------------------------------------------------------------------

5. Get details of single vendor

--------------------------------------------------------------------------------

Method : GET

URL : /api/vendor/get/one/:vendorID

--------------------------------------------------------------------------------

6. Update single vendor

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/patch

Input : {

\_id : objectID,

companyName : String,

groupName : String,

CIN : String,

COI : Array,

TAN : String,

companyLogo : Array,

website : String,

companyPhone : String,

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Add vendor location

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/patch/addLocation

Input : {

\_id : objectID,

locationType : String,

branchCode : Number,

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

GSTIN : String,

GSTDocument : Array,

PAN : String,

PANDocument : Array

}

Output :

If successfully

{

created : true,

}

If not

{ created : false }

--------------------------------------------------------------------------------

8. Get details of single location

--------------------------------------------------------------------------------

Method : POST

URL : /api/vendor/post/singleLocation

Input : {

entityID : objectID,

locationID : ObjectID

}

--------------------------------------------------------------------------------

9. Update single location

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/patch/updateSingleLocation

Inut : {

entityID : objectID,

locationID : ObjectID,

locationType : String,

branchCode : Number,

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

GSTIN : String,

GSTDocument : Array,

PAN : String,

PANDocument : Array

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

10. Add vendor contact person

-----------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/patch/addContact

Inut : {

entityID : objectID,

branchCode : Number,

name : String,

phone : String,

altPhone : String,

email : String,

department : String,

designation : String,

employeeID : String

}

Output :

If successfully

{

created : true,

}

If not

{ created : false }

--------------------------------------------------------------------------------

11. Get details of single contact person

--------------------------------------------------------------------------------

Method : POST

URL : /api/vendor/post/singleContact

Input : {

entityID : objectID,

contactID : ObjectID

}

--------------------------------------------------------------------------------

12. Update single contact person

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/patch/updateSingleContact

Inut : {

entityID : objectID,

contactID : ObjectID,

branchCode : Number,

name : String,

phone : String,

altPhone : String,

email : String,

department : String,

designation : String,

employeeID : String

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

13. Delete single supplier

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/vendor/delete/:vendorID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

--------------------------------------------------------------------------------

14. Delete single location for vendor

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/delete/:vendorID/:locationID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

--------------------------------------------------------------------------------

15. Delete single contact person for vendor

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vendor/delete/:vendorID/:contactID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

# 4.4 Employee Master

|  |  |
| --- | --- |
| **Employee Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| vendorId | { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' } |
| type | String |
| firstName | String |
| middleName | String |
| lastName | String |
| DOB | Date |
| gender | String |
| contactNo | String |
| altContactNo | String |
| profilePhoto | String |
| email | String |
| whatsappNo | String |
| BadgeNumber | String |
| verificationNumber | String |
| departmentId | { type: mongoose.Schema.Types.ObjectId, ref: 'departmentmasters' } |
| designationId | { type: mongoose.Schema.Types.ObjectId, ref: 'designationmasters' } |
| employeeId | String |
| bookingApprovalRequired | String |
| loginCredential | String |
| approvingAuthorityId1 | String |
| approvingAuthorityId2 | String |
| approvingAuthorityId3 | String |
| preApprovedParameter | String |
| preApprovedParameterValue | Number |
| **address** | Array |
| addressLine1 | String |
| addressLine2 | String |
| landmark | String |
| area | String |
| city | String |
| district | String |
| stateCode | String |
| state | String |
| countryCode | String |
| country | String |
| pincode | Number |
| latitude | String |
| longitude | String |
| addressProof | Array |
| **address** | Array |
| licenseNo | String |
| effectiveTo | Date |
| licenseProof | Array |
| **identityProof** | Array |
| **verificationProof** | Array |
| **aadhar** | Array |
| aadharNo | String |
| aadharProof | Array |
| corporateId | { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' } |
| userId | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| status | String |
| fileName | String |
| createdBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new employee | Completed |
| 2. List of all employees | Completed |
| 3. Count of all employees | Completed |
| 4. filter employees | Completed |
| 5. Get details of single employee | Completed |
| 6. Update single employee | Completed |
| 7. Search employee | Completed |
| 8. Delete single employee | Completed |

--------------------------------------------------------------------------------

1. Insert new **employee**

--------------------------------------------------------------------------------

Method : POST

URL : /api/employee/post

Input :

{

\_id : mongoose.Schema.Types.ObjectId,

vendorId : { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' },

type : String,

firstName : String,

middleName : String,

lastName : String,

DOB : Date,

gender : String,

contactNo : String,

altContactNo : String,

profilePhoto : String,

email : String,

whatsappNo : String,

workLocation : String,

batchNumber : String,

verificationNumber : String,

designationId : { type: mongoose.Schema.Types.ObjectId, ref: 'designationmasters' },

departmentId : { type: mongoose.Schema.Types.ObjectId, ref: 'departmentmasters' },

employeeId : String,

bookingApprovalRequired : String,

loginCredential : String,

approvingAuthorityId1 : String,

approvingAuthorityId2 : String,

approvingAuthorityId3 : String,

preApprovedParameter : String,

preApprovedParameterValue : Number,

address : [{

addressLine1 : String,

addressLine2 : String,

landmark : String,

area : String,

city : String,

district : String,

stateCode : String,

state : String,

countryCode : String,

country : String,

pincode : Number,

latitude : String,

longitude : String,

addressProof : Array

}],

drivingLicense :{

licenseNo : String,

effectiveTo : Date,

licenseProof : Array

},

identityProof : Array,

verificationProof : Array,

aadhar :{

aadharNo : String,

aadharProof : Array

},

corporateId : { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' },

userId : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

status : String,

fileName : String,

createdBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

createdAt : Date,

updateLog : [

{

updatedAt : Date,

updatedBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

]

}

Output :

If successfully

{

created : true,

ID : String

}

--------------------------------------------------------------------------------

2. List of all employees

--------------------------------------------------------------------------------

Method : GET

URL : /api/employees/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all employees

--------------------------------------------------------------------------------

Method : GET

URL : /api/employees/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter employees

--------------------------------------------------------------------------------

Method : POST

URL : /api/employees/get/filterEntities

Input : {

department : array of department ids ,

designation : array of designation ids,

initial : “A”,

stateCodes : [“MH”,”AP”]

districts : [“Pune”, “Mumbai”]

}

--------------------------------------------------------------------------------

5. Get details of single employee

--------------------------------------------------------------------------------

Method : GET

URL : /api/employees/get/one/:employeeID

--------------------------------------------------------------------------------

6. Update single employee

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/employees/patch

Input : {

\_id : objectID,

firstName : String,

middleName : String,

lastName : String,

DOB : Date,

gender : String,

contactNo : String,

altContactNo : String,

profilePhoto : Array,

whatsappNo : String,

designation : String,

department : String,

batchNumber : String,

verificationNumber : String,

designationId : { type: mongoose.Schema.Types.ObjectId, ref: 'designationmasters' },

departmentId : { type: mongoose.Schema.Types.ObjectId, ref: 'departmentmasters' },

employeeId : String,

bookingApprovalRequired : String,

loginCredential : String,

approvingAuthorityId1 : String,

approvingAuthorityId2 : String,

approvingAuthorityId3 : String,

preApprovedParameter : String,

preApprovedParameterValue : Number,

address : [{

addressLine1 : String,

addressLine2 : String,

landmark : String,

area : String,

city : String,

district : String,

stateCode : String,

state : String,

countryCode : String,

country : String,

pincode : Number,

latitude : String,

longitude : String,

addressProof : Array

}],

drivingLicense :{

licenseNo : String,

effectiveTo : Date,

licenseProof : Array

},

identityProof : Array,

verificationProof : Array,

aadhar :{

aadharNo : String,

aadharProof : Array

},

corporateId : { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' },

userId : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

status : String,

fileName : String,

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Search employees

--------------------------------------------------------------------------------

Method : GET

URL : /api/employees/search/:string

--------------------------------------------------------------------------------

8. Delete single employee

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/employees/delete/:employeeID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

# 4.5 Driver Master

|  |  |
| --- | --- |
| **Driver Master** |  |
| **Field Name** | **Field Type** |
| \_id | ObjectID |
| firstName | String |
| middleName | String |
| lastName | String |
| DOB | Date |
| gender | String |
| contactNo | String |
| altContactNo | String |
| profilePhoto | String |
| email | String |
| whatsappNo | String |
| department | String |
| designation | String |
| **address** | **Array** |
| addressLine1 | String |
| addressLine2 | String |
| landmark | String |
| area | String |
| city | String |
| district | String |
| stateCode | String |
| state | String |
| countryCode | String |
| country | String |
| pincode | Number |
| latitude | String |
| longitude | String |
| **drivingLicense** | Array |
| licenseNo | String |
| effectiveFrom | Date |
| effectiveTo | String |
| **licenseProof** | Array |
| **pan** | Array |
| PAN | String |
| PANProof | Array |
| **aadharProof** | String |
| aadharNo | String |
| aadharProof | Array |
| **voterID** | Array |
| voterID | String |
| **voterIDProof** | Array |
| **passport** | Array |
| passortNo | String |
| **passportProof** | **Array** |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new driver | Completed |
| 2. List of all drivers | Completed |
| 3. Count of all drivers | Completed |
| 4. filter drivers | Completed |
| 5. Get details of single driver | Completed |
| 6. Update single driver | Completed |
| 7. Search driver | Completed |
| 8. Delete single driver | Completed |

--------------------------------------------------------------------------------

1. Insert new driver

--------------------------------------------------------------------------------

Method : POST

URL : /api/driver/post

Input :

{

firstName : String,

middleName : String,

lastName : String,

DOB : Date,

gender : String,

contactNo : String,

altContactNo : String,

profilePhoto : Array,

email : String,

whatsappNo : String,

designation : String,

department : String,

address : [{

addressLine1 : String,

addressLine2 : String,

landmark : String,

area : String,

city : String,

district : String,

stateCode : String,

state : String,

countryCode : String,

country : String,

pincode : Number,

latitude : String,

longitude : String,

}],

drivingLicense : [{

licenseNo : String,

effectiveTo : Date,

licenseProof : Array

}],

pan : [{

PAN : String,

PANProof : Array

}],

aadhar : [{

aadharNo : String,

aadharProof : Array

}],

voterID : [{

voterID : String,

voterIDProof : Array

}],

passport : [{

passportNo : String,

passportProof : Array

}],

createdBy : { type: mongoose.Schema.Types.ObjectId, ref:'users' },

createdAt : Date,

}

Output :

If successfully

{

created : true,

ID : String

}

--------------------------------------------------------------------------------

2. List of all drivers

--------------------------------------------------------------------------------

Method : GET

URL : /api/drivers/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all drivers

--------------------------------------------------------------------------------

Method : GET

URL : /api/drivers/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter drivers

--------------------------------------------------------------------------------

Method : POST

URL : /api/drivers/get/filterEntities

Input : {

stateCodes : [“MH”,”AP”]

districts : [“Pune”, “Mumbai”],

initial : “A”

}

--------------------------------------------------------------------------------

5. Get details of single driver

--------------------------------------------------------------------------------

Method : GET

URL : /api/drivers/get/one/:employeeID

--------------------------------------------------------------------------------

6. Update single driver

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/drivers/patch

Input : {

\_id : objectID,

firstName : String,

middleName : String,

lastName : String,

DOB : Date,

gender : String,

contactNo : String,

altContactNo : String,

profilePhoto : Array,

whatsappNo : String,

designation : String,

department : String,

address : [{

addressLine1 : String,

addressLine2 : String,

landmark : String,

area : String,

city : String,

district : String,

stateCode : String,

state : String,

countryCode : String,

country : String,

pincode : Number,

latitude : String,

longitude : String,

}],

drivingLicense : [{

licenseNo : String,

effectiveTo : Date,

licenseProof : Array

}],

pan : [{

PAN : String,

PANProof : Array

}],

aadhar : [{

aadharNo : String,

aadharProof : Array

}],

voterID : [{

voterID : String,

voterIDProof : Array

}],

passport : [{

passportNo : String,

passportProof : Array

}],

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Search driver

--------------------------------------------------------------------------------

Method : GET

URL : /api/drivers/search/:string

--------------------------------------------------------------------------------

8. Delete single driver

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/drivers/delete/:driverID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

# 4.6 Guest Master

|  |  |
| --- | --- |
| **Guest Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| firstName | String |
| middleName | String |
| lastName | String |
| gender | String |
| contactNo | String |
| email | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new guest | Completed |
| 2. List of all guest | Completed |
| 3. Count of all guest | Completed |
| 4. filter guest | Completed |
| 5. Get details of single guest | Completed |
| 6. Update single guest | Completed |
| 7. Search guest | Completed |
| 8. Delete single guest | Completed |

--------------------------------------------------------------------------------

1. Insert new guest

--------------------------------------------------------------------------------

Method : POST

URL : /api/guest/post

Input :

{

firstName : String,

middleName : String,

lastName : String,

gender : String,

contactNo : String,

profilePhoto : Array,

email : String,

whatsappNo : String,

designation : String,

department : String,

createdBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

createdAt : Date,

}

Output :

If successfully

{

created : true,

ID : String

}

--------------------------------------------------------------------------------

2. List of all guest

--------------------------------------------------------------------------------

Method : GET

URL : /api/guest/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all guest

--------------------------------------------------------------------------------

Method : GET

URL : /api/guest/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter guest

--------------------------------------------------------------------------------

Method : POST

URL : /api/guest/get/filterEntities

Input : {

department : array of department ids ,

designation : array of designation ids,

initial : “A”,

stateCodes : [“MH”,”AP”]

districts : [“Pune”, “Mumbai”]

}

--------------------------------------------------------------------------------

5. Get details of single guest

--------------------------------------------------------------------------------

Method : GET

URL : /api/guest/get/one/:employeeID

--------------------------------------------------------------------------------

6. Update single guest

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/guest/patch

Input : {

\_id : objectID,

firstName : String,

middleName : String,

lastName : String,

gender : String,

contactNo : String,

profilePhoto : Array,

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Search guest

--------------------------------------------------------------------------------

Method : GET

URL : /api/guest/search/:string

--------------------------------------------------------------------------------

8. Delete single guest

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/guest/delete/:guestID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

# 4.7. Category Master

|  |  |
| --- | --- |
| **Category Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| category | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new Category | Completed |
| 2. Details of One Category | Completed |
| 3. Count of Categories | Completed |
| 4. List details of all the Categories | Completed |
| 5. Update Category | Completed |
| 6. Search Category | Completed |
| 7. Delete particular Category | Completed |

--------------------------------------------------------------------------------

1. Insert new **Category**

--------------------------------------------------------------------------------

Method : POST

URL : /api/categorymaster/post

Input :

{

"fieldValue" : "Petrol",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One Category

--------------------------------------------------------------------------------

Method : GET

URL : api/categorymaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"category": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of Categories

--------------------------------------------------------------------------------

Method : GET

URL : api/categorymaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the Categories

--------------------------------------------------------------------------------

Method : POST

URL : api/categorymaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": "5e31587a211dd0c3b8cc8692",

"locationType": "",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update Category

--------------------------------------------------------------------------------

Method : PATCH

URL : api/categorymaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search Category

--------------------------------------------------------------------------------

Method : GET

URL : api/categorymaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"category": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete Category

--------------------------------------------------------------------------------

Method : DELETE

URL : api/categorymaster/delete/:fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.8. Brand Master

|  |  |
| --- | --- |
| **Brand Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| brand | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new Brand | Completed |
| 2. Details of One Brand | Completed |
| **3. Count of Brands** | Completed |
| 4. List details of all the Brands | Completed |
| 5. Update Brand | Completed |
| 6. Search Brand | Completed |
| 7. Delete particular Brand | Completed |

--------------------------------------------------------------------------------

1. Insert new **brand**

--------------------------------------------------------------------------------

Method : POST

URL : /api/brandmaster/post

Input :

{

"fieldValue" : "TATA",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One Brand

--------------------------------------------------------------------------------

Method : GET

URL : api/brandmaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"category": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of Brands

--------------------------------------------------------------------------------

Method : GET

URL : api/brandmaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the Brands

--------------------------------------------------------------------------------

Method : POST

URL : api/brandmaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": "5e31587a211dd0c3b8cc8692",

"locationType": "",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update Brand

--------------------------------------------------------------------------------

Method : PATCH

URL : api/brandmaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search Brand

--------------------------------------------------------------------------------

Method : GET

URL : api/brandmaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"brand": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete Brand

--------------------------------------------------------------------------------

Method : DELETE

URL : api/brandmaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.9. Fuel Type Master

|  |  |
| --- | --- |
| **Fuel Type Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| fuelType | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new fuel type | Completed |
| 2. Details of One fuel type | Completed |
| 3. Count of fuel type | Completed |
| 4. List details of all the fuel type | Completed |
| 5. Update fuel type | Completed |
| 6. Search fuel type | Completed |
| 7. Delete particular fuel type | Completed |

--------------------------------------------------------------------------------

1. Insert new fuel type

--------------------------------------------------------------------------------

Method : POST

URL : /api/fueltypemaster/post

Input :

{

"fieldValue" : "test",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One fuelType

--------------------------------------------------------------------------------

Method : GET

URL : api/fueltypemaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"fuelType": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of fuelType

--------------------------------------------------------------------------------

Method : GET

URL : api/fueltypemaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the fueltype

--------------------------------------------------------------------------------

Method : POST

URL : api/fueltypemaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": “5e31587a211dd0c3b8cc8692",

"fuelType": "test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update fuelType

--------------------------------------------------------------------------------

Method : PATCH

URL : api/fueltypemaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search fuel type

--------------------------------------------------------------------------------

Method : GET

URL : api/fueltypemaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"designation": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete fuel type

--------------------------------------------------------------------------------

Method : DELETE

URL : api/fueltypemaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.10. Model Master

|  |  |
| --- | --- |
| **Model Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| brandId | { type: mongoose.Schema.Types.ObjectId, ref: 'brandmasters' } |
| model | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new model | Completed |
| 2. Details of One model | Completed |
| 3. Count of models | Completed |
| 4. List details of all the models | Completed |
| 5. Update model | Completed |
| 6. Search models | Completed |
| 7. Delete particular model | Completed |

--------------------------------------------------------------------------------

1. Insert new model

--------------------------------------------------------------------------------

Method : POST

URL : /api/modelmaster/post

Input :

{

"fieldValue" : "test",

“dropdownID” : ObjectID from brandmasters,

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of one Model

--------------------------------------------------------------------------------

Method : GET

URL : api/modelmaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

“brandId” : “5e2ed7976a0bb6411a8e6c11”

"model": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of models

--------------------------------------------------------------------------------

Method : GET

URL : api/modelmaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List of all the models

--------------------------------------------------------------------------------

Method : POST

URL : api/brandmaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": "5e31587a211dd0c3b8cc8692",

“brandId” : “5e2ed7976a0bb6411a8e6c11”

"model": "Test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update model

--------------------------------------------------------------------------------

Method : PATCH

URL : api/modelmaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“dropdownID” : ObjectID from brandmasters,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search model

--------------------------------------------------------------------------------

Method : GET

URL : api/modelmaster/search/:string

--------------------------------------------------------------------------------

7. Delete model

--------------------------------------------------------------------------------

Method : DELETE

URL : api/modelmaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.11. Department Master

|  |  |
| --- | --- |
| **Department Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| department | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new department | Completed |
| 2. Details of One department | Completed |
| 3. Count of departments | Completed |
| 4. List details of all the departments | Completed |
| 5. Update department | Completed |
| 6. Search department | Completed |
| 7. Delete particular department | Completed |

--------------------------------------------------------------------------------

1. Insert new **department**

--------------------------------------------------------------------------------

Method : POST

URL : /api/departmentmaster/post

Input :

{

"fieldValue" : "test",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One Category

--------------------------------------------------------------------------------

Method : GET

URL : api/departmentmaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"department": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of Categories

--------------------------------------------------------------------------------

Method : GET

URL : api/departmentmaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the departments

--------------------------------------------------------------------------------

Method : POST

URL : api/brandmaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": "5e31587a211dd0c3b8cc8692",

"department": "test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update department

--------------------------------------------------------------------------------

Method : PATCH

URL : api/departmentmaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search department

--------------------------------------------------------------------------------

Method : GET

URL : api/departmentmaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"department": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete department

--------------------------------------------------------------------------------

Method : DELETE

URL : api/departmentmaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.12. Designation Master

|  |  |
| --- | --- |
| **DesignationMaster** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| designation | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new designation | Completed |
| 2. Details of One designation | Completed |
| 3. Count of designation | Completed |
| 4. List details of all the designations | Completed |
| 5. Update designation | Completed |
| 6. Search designation | Completed |
| 7. Delete particular designation | Completed |

--------------------------------------------------------------------------------

1. Insert new designation

--------------------------------------------------------------------------------

Method : POST

URL : /api/designationmaster/post

Input :

{

"fieldValue" : "test",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One designation

--------------------------------------------------------------------------------

Method : GET

URL : api/designationmaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"designation": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of designations

--------------------------------------------------------------------------------

Method : GET

URL : api/designationmaster/get/count

Output :

{

  "count": 10

}

--------------------------------------------------------------------------------

4. List details of all the departments

--------------------------------------------------------------------------------

Method : POST

URL : api/designationmaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": "5e31587a211dd0c3b8cc8692",

"designation": "test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": 0"5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update designation

--------------------------------------------------------------------------------

Method : PATCH

URL : api/designationmaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search designation

--------------------------------------------------------------------------------

Method : DELETE

URL : api/designationmaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"designation": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete designation

--------------------------------------------------------------------------------

Method : DELETE

URL : api/designationmaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.13. Package Type Master

|  |  |
| --- | --- |
| **Package Type Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| packageType | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new package type | Completed |
| 2. Details of One package type | Completed |
| 3. Count of package type | Completed |
| 4. List details of all the package type | Completed |
| 5. Update package type | Completed |
| 6. Search package type | Completed |
| 7. Delete particular package type | Completed |

--------------------------------------------------------------------------------

1. Insert new package type

--------------------------------------------------------------------------------

Method : POST

URL : /api/packagetypemaster/post

Input :

{

"fieldValue" : "test",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One packageType

--------------------------------------------------------------------------------

Method : GET

URL : api/packagetypemaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"fuelType": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of packageType

--------------------------------------------------------------------------------

Method : GET

URL : api/packagetypemaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the packagetype

--------------------------------------------------------------------------------

Method : POST

URL : api/packagetypemaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": “5e31587a211dd0c3b8cc8692",

"packageType": "test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update packageType

--------------------------------------------------------------------------------

Method : PATCH

URL : api/packagetypemaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search package type

--------------------------------------------------------------------------------

Method : GET

URL : api/packagetypemaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"packageType": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete package type

--------------------------------------------------------------------------------

Method : DELETE

URL : api/packagetypemaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.14. Location Type Master

|  |  |
| --- | --- |
| **Location Type Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| locationType | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new location type | Completed |
| 2. Details of One location type | Completed |
| 3. Count of location type | Completed |
| 4. List details of all the location type | Completed |
| 5. Update location type | Completed |
| 6. Search location type | Completed |
| 7. Delete particular location type | Completed |

--------------------------------------------------------------------------------

1. Insert new location type

--------------------------------------------------------------------------------

Method : POST

URL : /api/locationtypemaster/post

Input :

{

"fieldValue" : "test",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One locationType

--------------------------------------------------------------------------------

Method : GET

URL : api/locationtypemaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"locationType": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of locationType

--------------------------------------------------------------------------------

Method : GET

URL : api/locationtypemaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the locationtype

--------------------------------------------------------------------------------

Method : POST

URL : api/locationtypemaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": “5e31587a211dd0c3b8cc8692",

"locationType": "test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update locationType

--------------------------------------------------------------------------------

Method : PATCH

URL : api/locationtypemaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search package type

--------------------------------------------------------------------------------

Method : GET

URL : api/locationtypemaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"locationType": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete location type

--------------------------------------------------------------------------------

Method : DELETE

URL : api/locationtypemaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.15. Module Master

|  |  |
| --- | --- |
| **Module Type Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| moduleName | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new module | Completed |
| 2. Details of One l module | Completed |
| 3. Count of modules | Completed |
| 4. List details of all the modules | Completed |
| 5. Update module | Completed |
| 6. Search module | Completed |
| 7. Delete particular module | Completed |

--------------------------------------------------------------------------------

1. Insert new module

--------------------------------------------------------------------------------

Method : POST

URL : /api/modulemaster/post

Input :

{

"fieldValue" : "test",

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of One module

--------------------------------------------------------------------------------

Method : GET

URL : api/modulemaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"moduleName": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of modules

--------------------------------------------------------------------------------

Method : GET

URL : api/modulemaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List details of all the modules

--------------------------------------------------------------------------------

Method : POST

URL : api/modulemaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": “5e31587a211dd0c3b8cc8692",

"moduleName": "test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update module

--------------------------------------------------------------------------------

Method : PATCH

URL : api/modulemaster/patch

Input : { “fieldID” :5e2ed7ac6a0bb6411a8e6c20,

“fieldValue”:newwww,

“updatedBy” :5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search modules

--------------------------------------------------------------------------------

Method : GET

URL : api/modulemaster/search/:string

Output :

[

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

"moduleName": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": [],

"\_\_v": 0

}

]

--------------------------------------------------------------------------------

7. Delete module

--------------------------------------------------------------------------------

Method : DELETE

URL : api/modulemaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.16. Facilities Master

|  |  |
| --- | --- |
| **Facility Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| moduleId | { type: mongoose.Schema.Types.ObjectId, ref: 'modulemasters' } |
| facility | String |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new facility | Completed |
| 2. Details of One facility | Completed |
| 3. Count of facilities | Completed |
| 4. List details of all the facilities | Completed |
| 5. Update facility | Completed |
| 6. Search facilities | Completed |
| 7. Delete particular facility | Completed |

--------------------------------------------------------------------------------

1. Insert new facility

--------------------------------------------------------------------------------

Method : POST

URL : /api/facilitymaster/post

Input :

{

"fieldValue" : "test",

“dropdownID” : ObjectID from modulemasters,

"createdBy" : "5e1c089b31609a29fdaad63f"

}

Output :

If successfully

{

"created": true,

"fieldID": "5e31680ce1563b2866cd48ef"

}

If duplicated

{

"duplicated": true

}

--------------------------------------------------------------------------------

2. Details of one facility

--------------------------------------------------------------------------------

Method : GET

URL : api/facilitymaster/get/one/5e2fd71b0074035dff3aab22

Output :

{

"\_id": "5e2ed7976a0bb6411a8e6c1f",

“moduleId” : “5e2ed7976a0bb6411a8e6c11”

"facility": "Test",

"createdBy": "5e1c089b31609a29fdaad63f",

"createdAt": "2020-01-27T12:29:11.719Z",

"updateLog": []

}

--------------------------------------------------------------------------------

3. Count of facilities

--------------------------------------------------------------------------------

Method : GET

URL : api/facilitymaster/get/count

Output :

{

"count": 10

}

--------------------------------------------------------------------------------

4. List of all the facilities

--------------------------------------------------------------------------------

Method : POST

URL : api/facilitymaster/get/list

Input : {"startRange":0,"limitRange":10}

Output :

[{

"\_id": "5e31587a211dd0c3b8cc8692",

“moduleId” : “5e2ed7976a0bb6411a8e6c11”

"facility": "Test",

"createdBy": "5e2537d0b279954b3f6015f9",

"createdAt": "2020-01-29T10:03:38.451Z",

"updateLog": [ {

"\_id": "5e3166c2211dd0c3b8cc8698",

"updatedAt": "2020-01-29T11:04:34.172Z",

"updatedBy": "5e2537d0b279954b3f6015f9"

} ]

}]

--------------------------------------------------------------------------------

5. Update facility

--------------------------------------------------------------------------------

Method : PATCH

URL : api/facilitymaster/patch

Input : { “fieldID” : 5e2ed7ac6a0bb6411a8e6c20,

“dropdownID” : ObjectID from modulemasters

“fieldValue” : newwww,

“updatedBy” : 5e1c089b31609a29fdaad63f }

Output :

If successfully

{ "updated": true }

If not updated

{ "updated": false }

--------------------------------------------------------------------------------

7. Search model

--------------------------------------------------------------------------------

Method : GET

URL : api/facilitymaster/search/:string

--------------------------------------------------------------------------------

7. Delete model

--------------------------------------------------------------------------------

Method : DELETE

URL : api/facilitymaster/delete/: fieldID

Output :

If successfully

{ "deleted" : true }

If not updated

{ "deleted" : false }

# 4.17 Vehicle Master

|  |  |
| --- | --- |
| **Vehicle Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| categoryId | { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' } |
| category | String |
| brandId | { type: mongoose.Schema.Types.ObjectId, ref: 'brandmasters' } |
| brand | String |
| modelId | { type: mongoose.Schema.Types.ObjectId, ref: 'modelmasters' } |
| model | String |
| capacity | String |
| fuelTypeId | { type: mongoose.Schema.Types.ObjectId, ref: 'fueltypemasters' } |
| fuelType | String |
| vehicleDriveType | String |
| ownership | String |
| vehicleNumber | String |
| registrationDate | Date |
| RCDoc | Array |
| insuranceDate | Date |
| insuranceDoc | Array |
| permitType | String |
| permitValidUpto | Date |
| permitDoc | Array |
| authorizationUpto | Date |
| authorizationDoc | Array |
| PUCValidUpto | Date |
| PUCDoc | Array |
| createdBy | String |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | new Date() |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new vehicle | Completed |
| 2. List of all vehicles | Completed |
| 3. Count of all vehicles | Completed |
| 4. filter vehicle | Completed |
| 5. Get details of single vehicle | Completed |
| 6. Update single vehicle | Completed |
| 7. Search vehicle | Completed |
| 8. Delete single vehicle | Completed |

--------------------------------------------------------------------------------

1. Insert new vehicle

--------------------------------------------------------------------------------

Method : POST

URL : /api/vehiclemaster/post

Input :

{

categoryId : { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' },

category : String,

brandId : { type: mongoose.Schema.Types.ObjectId, ref: 'brandmasters' },

brand : String,

modelId : { type: mongoose.Schema.Types.ObjectId, ref: 'modelmasters' },

model : String,

capacity : String,

fuelTypeId : { type: mongoose.Schema.Types.ObjectId, ref: 'fueltypemasters' },

fuelType : String,

vehicleDriveType : String,

ownership : String,

vehicleNumber : String,

registrationDate : Date,

RCDoc : Array,

insuranceDate : Date,

insuranceDoc : Array,

permitType : String,

permitValidUpto : Date,

permitDoc : Array,

authorizationUpto : Date,

authorizationDoc : Array,

PUCValidUpto : Date,

PUCDoc : Array,

createdBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

createdAt : new Date()

}

Output :

If successfully

{

created : true,

ID : String

}

--------------------------------------------------------------------------------

2. List of all vehicle

--------------------------------------------------------------------------------

Method : GET

URL : /api/vehiclemaster/get

Input : - NA-

--------------------------------------------------------------------------------

3. Count of all vehicles

--------------------------------------------------------------------------------

Method : GET

URL : /api/vehiclemaster/get/count

Input : - NA-

--------------------------------------------------------------------------------

4. filter vehicle

--------------------------------------------------------------------------------

Method : POST

URL : /api/vehiclemaster/post/list/filterVehicles

Input : {

categoryId : array of category ids ,

brandId : array of brand ids,

modelId : array of model ids,

fuelTypeId : array of fuelType ids

}

--------------------------------------------------------------------------------

5. Get details of single guest

--------------------------------------------------------------------------------

Method : GET

URL : /api/vehiclemaster/get/one/:vehicleID

--------------------------------------------------------------------------------

6. Update single vehicle

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/vehiclemaster/patch

Input : {

\_id : objectID,

categoryId : { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' },

category : String,

brandId : { type: mongoose.Schema.Types.ObjectId, ref: 'brandmasters' },

brand : String,

modelId : { type: mongoose.Schema.Types.ObjectId, ref: 'modelmasters' },

model : String,

capacity : String,

fuelTypeId : { type: mongoose.Schema.Types.ObjectId, ref: 'fueltypemasters' },

fuelType : String,

vehicleDriveType : String,

ownership : String,

vehicleNumber : String,

registrationDate : Date,

RCDoc : Array,

insuranceDate : Date,

insuranceDoc : Array,

permitType : String,

permitValidUpto : Date,

permitDoc : Array,

authorizationUpto : Date,

authorizationDoc : Array,

PUCValidUpto : Date,

PUCDoc : Array,

updatedAt : Date,

updatedBy :{ type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

Output :

If successfully

{

updated : true,

}

If not

{ updated : false }

--------------------------------------------------------------------------------

7. Search vehicle

--------------------------------------------------------------------------------

Method : GET

URL : /api/vehiclemaster/search/:string

--------------------------------------------------------------------------------

8. Delete single vehicle

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/vehiclemaster/delete/:vehicleID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

# 4.18. Package Master

|  |  |
| --- | --- |
| **Package Master** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| packageTypeId | { type: mongoose.Schema.Types.ObjectId, ref: 'packagetypemasters' } |
| packageName | String |
| fixCharges | Number |
| maxHours | Number |
| maxKm | Number |
| createdBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | Date |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert new package | Completed |
| 2. Details of One package | Completed |
| 3. Count of packages | Completed |
| 4. List details of all the packages | Completed |
| 5. Update package | Completed |
| 6. Delete package | Completed |

--------------------------------------------------------------------------------

1. Insert new package

--------------------------------------------------------------------------------

Method : POST

URL : /api/packagemaster/post

Input :

{

packageTypeId : { type: mongoose.Schema.Types.ObjectId, ref: 'packagetypemasters' },

packageName : String,

fixCharges : Number,

maxHours : Number,

maxKm : Number,

createdBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

createdAt : Date }

Output :

If successfully

{

"created": true,

"packageId": "5e31680ce1563b2866cd48ef"

}

--------------------------------------------------------------------------------

2. Details of One package

--------------------------------------------------------------------------------

Method : GET

URL : /api/packagemaster/get/one/:packageID

--------------------------------------------------------------------------------

3. Count of all packages

--------------------------------------------------------------------------------

Method : GET

URL : /api/packagemaster/get/count

--------------------------------------------------------------------------------

4. List of all packages

--------------------------------------------------------------------------------

Method : GET

URL : /api/packagemaster/get

Input : - NA-

--------------------------------------------------------------------------------

5. Update package

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/packagemaster/get

Input : {

packageID : ObjectID,

packageTypeId : { type: mongoose.Schema.Types.ObjectId, ref: 'packagetypemasters' },

packageName : String,

fixCharges : Number,

maxHours : Number,

maxKm : Number

}

--------------------------------------------------------------------------------

6. Delete single package

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/packagemaster/delete/:packageID

If successfully

{

deleted : true,

}

If not

{ deleted : false }

# 5. Access Management

|  |  |
| --- | --- |
| **Access Management** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| role | { type: mongoose.Schema.Types.ObjectId, ref: 'packagetypemasters' } |
| **module** | **Array** |
| module | String |
| facility | String |
| createdBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | Date |
| updatedBy | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| 1. Insert access to roles | Completed |
| 2. Get all access with roles | Completed |
| 3. Get rolewise access | Completed |
| 4. Get rolewise access to given module | Completed |
| 5. Get access to facility of module | Completed |

--------------------------------------------------------------------------------

1. Insert access

--------------------------------------------------------------------------------

Method : POST

URL : /api/accessmaster/post

Input : {

role : "authuser",

module : array({module : "usermanagement", facility: "Add user"})

}

--------------------------------------------------------------------------------

2. Get all access with roles

--------------------------------------------------------------------------------

Method : GET

URL : /api/accessmaster/get

Output : {

"\_id" : ObjectId("5e55f9eaf8310e337056bca9"),

"role" : "admin",

"module" : [

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcb2"),

"module" : "Module Master"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcb1"),

"module" : "Module Master",

"facility" : "Create Module"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcb0"),

"module" : "Module Master",

"facility" : "Edit Module"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcaf"),

"module" : "Module Master",

"facility" : "Modules List "

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcae"),

"module" : "Corporate Master"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcad"),

"module" : "Corporate Master",

"facility" : "Create Corporate"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcac"),

"module" : "User Management"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcab"),

"module" : "User Management",

"facility" : "Create user"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcaa"),

"module" : "User Management",

"facility" : "Edit User"

}

],

"createdAt" : ISODate("2020-02-26T04:54:02.751Z"),

"updateLog" : []

--------------------------------------------------------------------------------

3.Get rolewise access

--------------------------------------------------------------------------------

Method : POST

URL : /api/accessmaster/getRolewiseAccess

Input : array - [“admin”, “superadmin”]

Output : {

"\_id" : ObjectId("5e55f9eaf8310e337056bca9"),

"role" : "admin",

"module" : [

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcb2"),

"module" : "Module Master"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcb1"),

"module" : "Module Master",

"facility" : "Create Module"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcb0"),

"module" : "Module Master",

"facility" : "Edit Module"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcaf"),

"module" : "Module Master",

"facility" : "Modules List "

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcae"),

"module" : "Corporate Master"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcad"),

"module" : "Corporate Master",

"facility" : "Create Corporate"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcac"),

"module" : "User Management"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcab"),

"module" : "User Management",

"facility" : "Create user"

},

{

"\_id" : ObjectId("5e55f9eaf8310e337056bcaa"),

"module" : "User Management",

"facility" : "Edit User"

}

],

"createdAt" : ISODate("2020-02-26T04:54:02.751Z"),

--------------------------------------------------------------------------------

4. Get rolewise access to given module

--------------------------------------------------------------------------------

Method : POST

URL : /api/accessmaster/getRolewiseAccessToModule

Input : {

roles: [“admin”, “superadmin”],

module : "Module Master"

}

Output :If have access {access:true}

If does not have access {access : false}

--------------------------------------------------------------------------------

5. Get access to facility of module

--------------------------------------------------------------------------------

Method : POST

URL : /api/accessmaster/getAccessToFacilityOfModule

Input : {

roles: [“admin”, “superadmin”],

module : "Module Master"

facility : "Create Module"

}

Output :If have access {access:true}

If does not have access {access : false}

# 6. Contract Management

|  |  |  |
| --- | --- | --- |
| **Contract Management** |  |  |
| **Field Name** | **Field Type** |  |
| \_id | PK |  |
| contractDate | Date |  |
| effectiveUpto | Date |  |
| corporateId | { type: mongoose.Schema.Types.ObjectId, ref: 'corporatemasters' } |  |
| corporateLocationId | String |  |
| vendorId | { type: mongoose.Schema.Types.ObjectId, ref: 'vendormasters' } |  |
| vendorLocationId | String |  |
| conditions | String |  |
| **packages** | **Array** |  |
| packageId | { type: mongoose.Schema.Types.ObjectId, ref: 'packagemasters' }, |  |
| packageName | String |  |
| MaxKm | Number |  |
| MaxHrs | Number |  |
| fixCharges | Number |  |
|  | **extras** | **Array** |
|  | categoryId | { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' } |
|  | category | String |
|  | extraHours | Number |
|  | extraKM | Number |
|  | graceHours | Number |
|  | graceKM | Number |
|  | driverAllowance | Number |
|  | nightCharges | Number |
|  | nightChargesFromTime | String |
|  | nightChargesToTime | String |
|  | nightChargesDay | String |
|  | earlyMorningCharges | Number |
|  | earlyMorningChargesFromTime | String |
|  | earlyMorningChargesToTime | String |
| createdBy |  | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt |  | Date |
| **updatedLog** |  | Array |
| updatedAt |  | Date |
| updatedBy |  | String |

**List of APIs**

|  |  |
| --- | --- |
| **API Name** | **Status of work** |
| **1. Create contract** | **Completed** |
| **2. Add packages to contract** | **Completed** |
| **3. Add terms and conditions to contract** | **Completed** |
| **4. Update Contract** | **Completed** |
| **5. Update single Package** | **Completed** |
| **6. Get list of contracts** | **Completed** |
| **7. Get count of contracts** | **Completed** |
| **8. Get details of single contract** | **Completed** |
| **9. Filter contract** | **Completed** |
| **10. Search Contract** | **Completed** |
| **11. Delete single contract** | **Completed** |
| **12. Delete single package in contract** | **Completed** |

--------------------------------------------------------------------------------

1. Create contract

--------------------------------------------------------------------------------

Method : POST

URL : /api/contract/post

Input : {

contractDate : Date,

effectiveUpto : Date,

corporateId : { type: mongoose.Schema.Types.ObjectId, ref:'corporatemasters' },

corporateLocationId : String,

vendorId : { type: mongoose.Schema.Types.ObjectId, ref: 'vendormasters' },

vendorLocationId : String,

}

If successfully

{

created : true,

ID : String

}

--------------------------------------------------------------------------------

2. Add packages to contract

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/contract/patch/addpackage

Input : {

packages : [

{

packageId : { type: mongoose.Schema.Types.ObjectId, ref: 'packagemasters' },

packageName : String,

MaxKm : Number,

MaxHrs : Number,

fixCharges : Number,

extras : [{

categoryId : { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' },

category : String,

extraHours : Number,

extraKM : Number,

graceHours : Number,

graceKM : Number,

driverAllowance : Number,

nightCharges : Number,

nightChargesFromTime : String,

nightChargesToTime : String,

nightChargesDay : String,

earlyMorningCharges : Number,

earlyMorningChargesFromTime : String,

earlyMorningChargesToTime : String,

}]

},

contractID : ObjectId

]

Output : If successfully

{

updated : true

}

If not { updated : false }

--------------------------------------------------------------------------------

3. Add terms and conditions to contract

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/contract/patch/addcondition

Input : {

packages : [

{

packageId : { type: mongoose.Schema.Types.ObjectId,

ref: 'packagemasters' },

packageName : String,

MaxKm : Number,

MaxHrs : Number,

fixCharges : Number,

extras : [{

categoryId : { type :mongoose.Schema.Types.ObjectId,

ref:'categorymasters' },

category : String,

extraHours : Number,

extraKM : Number,

graceHours : Number,

graceKM : Number,

driverAllowance : Number,

nightCharges : Number,

nightChargesFromTime : String,

nightChargesToTime : String,

nightChargesDay : String,

earlyMorningCharges : Number,

earlyMorningChargesFromTime : String,

earlyMorningChargesToTime : String,

}]

},

contractID : ObjectId

]

Output : If successfully

{

updated : true

}

If not { updated : false }

--------------------------------------------------------------------------------

4. Update Contract

------------------------------------------------------------------------------------------------------------------------

Method : PATCH

URL : /api/contract/patch

Input : {

\_id : ObjectID,

contractDate : Date,

effectiveUpto : Date,

corporateId : { type: mongoose.Schema.Types.ObjectId, ref: 'corporatemasters' },

corporateLocationId : String,

vendorId : { type: mongoose.Schema.Types.ObjectId, ref: 'vendormasters' },

vendorLocationId : String,

updatedBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}

Output : If successfully

{

updated : true

}

If not { updated : false }

--------------------------------------------------------------------------------

5. Update single Package

--------------------------------------------------------------------------------

Method : GET

URL : /api/contract/patch/updatepackage

Input : {

\_id : ObjectID,

packagesID : ObjectID,

packages : [

{

packageId : { type:mongoose.Schema.Types.ObjectId,

ref: 'packagemasters' },

packageName : String,

MaxKm : Number,

MaxHrs : Number,

fixCharges : Number,

extras : Array

} ]

}

Output : If successfully

{

updated : true

}

If not { updated : false }

--------------------------------------------------------------------------------

6. Get list of contracts

--------------------------------------------------------------------------------

Method : GET

URL : /api/contract/get/joincontractlist

-------------------------------------------------------------------------------

7. Get count of contracts

-------------------------------------------------------------------------------

Method : GET

URL : /api/contract/get/count

Output : {“count: 10 }

-------------------------------------------------------------------------------

8. Get details of single contract

-------------------------------------------------------------------------------

Method : GET

URL : /api/contract/get/joincontract/:contractID

--------------------------------------------------------------------------------

9. Filter contract

--------------------------------------------------------------------------------

Method : POST

URL : /api/contract/filterContract

Input : { corporateIds : array of object ids in corporate master,

vendorIds : array of object ids in vendor master,

stateCodes : array of stateCodes }

--------------------------------------------------------------------------------

10. Search Contract

--------------------------------------------------------------------------------

Method : POST

URL : /api/contract/search/:str

Input : { req.params.str }

--------------------------------------------------------------------------------

11. Delete single contract

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/contract/delete/:contractID

Input : { req.params.contractID }

Output : If successfully

{

deleted : true

}

If not { deleted : false }

--------------------------------------------------------------------------------

12.Delete single package in contract

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/contract/deletepackageincontract/:contractID/:packageID

Input : { req.params.contractID, req.params.packageID }

Output: If successfully

{

deleted : true

}

If not { deleted : false }

# 7. Booking Management

|  |  |
| --- | --- |
| **Booking Management** |  |
| **Field Name** | **Field Type** |
| \_id | PK |
| packageTypeId | { type: mongoose.Schema.Types.ObjectId, ref: packagetypemasters } |
| packageId | { type: mongoose.Schema.Types.ObjectId, ref: packagemasters } |
| tripType | String |
| **from** | **Array** |
| address | String |
| area | String |
| city | String |
| state | String |
| country | String |
| pincode | Number |
| latitude | Number |
| longitude | Number |
| **to** | **Array** |
| address | String |
| area | String |
| city | String |
| state | String |
| country | String |
| pincode | Number |
| latitude | Number |
| longitude | Number |
| pickupDate | Date |
| pickupTime | String |
| returnDate | Date |
| returnTime | String |
| specialInstruction | String |
| purposeOfTravel | String |
| reasonForSelectingVehicle | String |
| signature | String |
| vehicleCategoryId | { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' } |
| vehicleId | { type: mongoose.Schema.Types.ObjectId, ref: 'vehiclemasters' } |
| employeeId | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| managerId1 | { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' } |
| managerId2 | { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' } |
| managerId3 | { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' } |
| corporateId | { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' } |
| approvalRequired | String |
| estimatedCost | Number |
| **intermediateStops** | **Array** |
| address | String |
| area | String |
| city | String |
| state | String |
| country | String |
| pincode | Number |
| latitude | Number |
| longitude | Number |
| **status** | **Array** |
| value | String |
| statusBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| allocatedTo | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| allocatedToVendor | { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' } |
| statusAt | Date |
| remark | String |
| latitude | Number |
| longitude | Number |
| OTP | Number |
| odometerReading | Number |
| proof | String |
| statusValue | String |
| **routeCoordinates** | **Array** |
| latitude | Number |
| longitude | Number |
| distanceTravelled | Number |
| stop | Boolean |
| **tripExpenses** | Array |
| createdBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |
| createdAt | Date |
| **updatedLog** | Array |
| updatedAt | Date |
| updatedBy | { type: mongoose.Schema.Types.ObjectId, ref: 'users' } |

**List of APIs**

|  |  |
| --- | --- |
| API Name | Status of work |
| 1. Insert booking | Completed |
| 2. Get all bookings | Completed |
| 3. Get booking count | Completed |
| 4. Get details of single booking | Completed |
| 5. Update single booking | Completed |
| 6. Delete single booking | Completed |

--------------------------------------------------------------------------------

1. Insert booking

--------------------------------------------------------------------------------

Method : POST

URL : /api/bookingmaster/post

Input : {

packageTypeId : { type: mongoose.Schema.Types.ObjectId, ref: 'packagetypemasters' },

packageId : { type: mongoose.Schema.Types.ObjectId, ref: 'packagemasters' },

bookingId : Number,

tripType : String,

from : {

address : String,

area : String,

city : String,

state : String,

country : String,

pincode : String,

latitude : Number,

longitude : Number,

},

to : {

address : String,

area : String,

city : String,

state : String,

country : String,

pincode : String,

latitude : Number,

longitude : Number,

},

pickupDate : Date,

pickupTime : String,

returnDate : Date,

returnTime : String,

specialInstruction : String,

purposeOfTravel : String,

reasonForSelectingVehicle : String,

signature : String,

vehicleCategoryId : { type: mongoose.Schema.Types.ObjectId, ref: 'categorymasters' },

vehicleID : { type: mongoose.Schema.Types.ObjectId, ref: 'vehiclemasters' },

employeeId : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

corporateId : { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' },

managerId1 : { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' },

managerId2 : { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' },

managerId3 : { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' },

approvalRequired : String,

estimatedCost : Number,

intermediateStops : [{

address : String,

area : String,

city : String,

state : String,

country : String,

pincode : String,

latitude : String,

longitude : String

}],

status : [{

value : String,

statusBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

allocatedTo : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

allocatedToVendor : { type: mongoose.Schema.Types.ObjectId, ref: 'entitymasters' },

statusAt : Date,

remark : String,

latitude : Number,

longitude : Number,

OTP : Number,

odometerReading : Number,

proof : String,

}],

statusValue : String,

routeCoordinates : [{

latitude : Number,

longitude : Number,

distanceTravelled : Number,

stop : Boolean,

}],

tripExpenses : Array,

createdBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' },

createdAt : Date,

updateLog : [{

updatedAt : Date,

updatedBy : { type: mongoose.Schema.Types.ObjectId, ref: 'users' }

}]

}

Output: If successfully

{

created : true

}

If not { created : false }

--------------------------------------------------------------------------------

2. Get all bookings

--------------------------------------------------------------------------------

Method : GET

URL : /api/bookingmaster/get/list

--------------------------------------------------------------------------------

3. Get booking count

--------------------------------------------------------------------------------

Method : GET

URL : /api/bookingmaster/get/count

Output : { "count":10 }

--------------------------------------------------------------------------------

4. Get details of single booking

--------------------------------------------------------------------------------

Method : GET

URL : /api/bookingmaster/get/one/:bookingID

--------------------------------------------------------------------------------5. Update single booking

--------------------------------------------------------------------------------

Method : PATCH

URL : /api/bookingmaster/patch

Input : {

packageTypeId : { type:mongoose.Schema.Types.ObjectId,

ref: 'packagetypemasters' },

packageId : { type: mongoose.Schema.Types.ObjectId,

ref: 'packagemasters' },

tripType : String,

from : [{

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

latitude : Number,

longitude : Number,

}],

to : [{

addressLine1 : String,

addressLine2 : String,

countryCode : String,

country : String,

stateCode : String,

state : String,

district : String,

city : String,

area : String,

pincode : Number,

latitude : Number,

longitude : Number,

}],

pickupDate : Date,

pickupTime : String,

returnDate : Date,

returnTime : String,

vehicleCategoryId : { type: mongoose.Schema.Types.ObjectId,

ref: 'categorymasters' },

employeeId : { type: mongoose.Schema.Types.ObjectId,

ref: users' },

corporateId : { type: mongoose.Schema.Types.ObjectId,

ref: 'entitymasters' },

managerId1 : { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' },

managerId2 : { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' },

managerId3 : { type: mongoose.Schema.Types.ObjectId, ref: 'personmasters' },

approvalRequired : String,

estimatedCost : Number,

intermediateStops : [{

address : String,

area : String,

city : String,

state : String,

country : String,

pincode : String,

latitude : String,

longitude : String

}],

updatedBy : { type: mongoose.Schema.Types.ObjectId,

ref: 'users' }

}

Output: If successfully

{

updated : true

}

If not { updated : false }

--------------------------------------------------------------------------------6. Delete single booking

--------------------------------------------------------------------------------

Method : DELETE

URL : /api/bookingmaster/delete/:bookingID

Output: If successfully

{

deleted : true

}

If not { deleted : false }