**TimeFlyz Documentation - Backend**

**Login :** POST  /api/v1/auth/login

**Admin :**

email id + password

**Customer :** *//only customer can use phone number*

email id / mobile number + password

Check user is  active or not deleted

Check password by Md5 ,

store cookies by backend,

Store token in local storage, *// by frontend*

// Token generated by using user’s \_id and role (JWT)

**Logout :** GET  /api/v1/auth/logout

Only can logout when user is authorised

Remove cookies from backend,

Remove FCM token from database

Need to remove token from local storage *// by frontend*

—-----------------------------------------------------------------------------------------------------------------------------------------

**Forgot Password  :** POST  /api/v1/auth/forgot-password

Required fields :

const { email, phone } = *req*.body; *//only customer can use phone number*

Verify is there already OTP in the database with user credential  then update new OTP

If not :

Generate OTP and Send with Mobile number or Email

*\*\* send otp pending due to do not have credential*

Otp stored in optModel with expiry of 10 minutes

**Verify OTP  :** POST  /api/v1/auth/verify-otp

Required fields :                                    *//only customer can use phone number*

const { otp, email, phone } = *req*.body;

Get otp form OTP model, using email or phone number

Check token expiry                               *//10 min*

After verify otp delete it from database    *// for clear duplication*

Send success in response

**Set Forgot Password  :** POST  /api/v1/auth/set-forgot-password

Required fields :                                *// only customer can use phone number*

 const { email, password } = *req*.body;

Update password field in user document with encryption using Md5

Send success response

**Reset Password  :** POST  /api/v1/auth/reset-password

Required fields :               *// add confirm password from frontend side*

const { token, oldPassword, newPassword } = *req*.body;

Check old password using salt and Md5

If old password matches take new password , encrypt it and store in database

Send response

**Send Verification Email  :** POST  /api/v1/auth/sendEmailVerificationMail

*// pass token in header from frontend*

Take token from header

Split the user token from the frontend token

Attach token with verification link

Send link using “sendgrid” *// send email configuration pending*

Send success response

**Verify Email  :** POST  /api/v1/auth/verify-email

Required fields :

      const { token } = *req*.query;

Get token from query

Decode token with jwt

Find user from the decoded data

Update status of  “isEmailVerified” in database

Send success response

—-----------------------------------------------------------------------------------------------------------------------------------------

**Guest User :**

***// Create a new user***

**Create Guest User** **:** POST  /api/v1/guest/create

Used uuidv4 for create unique id for every guest user

Add new guest uuid, role = “guest”, type=”guest” in the payload

*Create token using this payload     // token expires in 24h*

*\*\*guest user data will not stored in database, it is only for visit purpose and searching for hotels”*

—-----------------------------------------------------------------------------------------------------------------------------------------

**Hotel  :**

***// Create a new hotel***

**Create** **:** POST  /api/v1/hotel/create

*\*\* Image upload pending*

Required fields :

    const {

      name,

      description,

      address,

      city,

      state,

      country,

      pincode,

      location,

      images,

      amenitiesId,

      adminId,

      Managers, //if manager already created

      contactNumber,

      email,

      website,

      rating,

      totalReviews,

    } = *req*.body;

NOTE : only Admin/ ChainAdmin and SuperAdmin can create Hotel

Chain admin can add multiple hotels but admin create only one hotel

Check if hotel exist with same name and with same pincode

Upload images of hotel to s3 cloud

Send success response

***// get hotel by id***

**Get Hotel By Id** **:** GET  /api/v1/hotel/get/:id

Required fields :

Hotel Id // in req.params.id

*// Only super admin can see inactive or deleted hotels*

***// delete hotel multiple by ids***

**Delete Hotel By Id** **:** DELETE  /api/v1/hotel/delete/:ids

Required fields :

Hotel Id // in req.params.ids

//can select multiple ids for delete multiple users

Only super admin can delete user

*\*\*only super admin can delete hotel*

***// update hotel by id***

**Update Hotel By Id** **:** PUT  /api/v1/hotel/update/:id

Required fields :

Hotel Id // in req.params.ids

Other fields that you want to change.

 const {

      name,

      description,

      address,

      city,

      state,

      country,

      pincode,

      location,

      images,

      amenitiesId,

      adminId,

      managers,

      contactNumber,

      email,

      website,

      isActive,

      isDeleted,

    } = *req*.body;

// only super admin can update inActive or deleted hotel

Check is new email already taken ?

*\*\*when user try to delete account it will not deleted but just update isDeleted field*

*\*\*only super admin can delete user profile*

***// search hotel***

**Search Hotel / Get All Hotels** **:** GET  /api/v1/hotel/getAll

*// Without Auth ( using guest user) or with login  
\*\* use guest login if user is not login*

Required fields :

      const { search = "", lat, lng, distance, adminId, customerId,

 amenitiesId } = *req*.query;

page, limit, skip *// for pagination*

      const page = parseInt(*req*.query.page) || 1;

      const limit = parseInt(*req*.query.limit);

User can search hotel by - distance ,hotel name, city, country, email, contact number  
User can search hotel by Amenities too*// pass amenitiesId from frontend*

*//only super admin can see inactive or deleted hotels*

Super Admin can search hotel by AdminIds  too *// pass adminId from frontend*

*\*\* only super admin and admin can search hotel by admin id*

—-----------------------------------------------------------------------------------------------------------------------------------------

**Hotel Amenities (only super admin) :**

***// Create a new hotel Amenities***

**Create Hotel Amenities** **:** POST  /api/v1/hotelAmenities/create

const { name} = *req*.body;

Check hotel amenity already exist with same name

***// get hotel Amenities by id***

**Get Hotel  Amenities By Id** **:** GET  /api/v1/hotelAmenities/get/:id

Required fields :

Hotel Id // in req.params.id

*// Only super admin can see inactive or deleted hotel amenities*

***// delete hotel Amenities by ids***

**Delete Hotel  Amenities By Ids** **:** DELETE  /api/v1/hotelAmenities/delete/:ids

Required fields :

Hotel Id // in req.params.ids

//can select multiple ids for delete multiple hotel amenities

Only super admin can delete hotel amenity

*\*\*only super admin can delete hotel amenity*

***// update hotel Amenities by id***

**Update Hotel  Amenities By Id** **:** PUT  /api/v1/hotelAmenities/update/:id

Required fields :

Hotel Id // in req.params.id

   const { name } = *req*.body;

Super admin can change names.

Check hotel amenity already exist with same name

***// Get all hotel Amenities***

**Get Hotel  Amenities** **:** GET  /api/v1/hotelAmenities/getAll

Required fields :

  const { search = "", hotelId } = *req*.query;

  const page = parseInt(*req*.query.page) || 1;

  const limit = parseInt(*req*.query.limit);

*// only super admin can see inactive or deleted hotels*

—-----------------------------------------------------------------------------------------------------------------------------------------

**User :**

***// Create a new user***

**Create User** **:** POST  /api/v1/user/create

const {

      name,

      email,

      Password, //confirm password field add in frontend

      role,enum:["superAdmin", "admin", "manager", "supportAgent"],

      phone,

      profilePic,

      hotelId,

      isActive,

      isChainAdmin,

    } = *req*.body;

*\*\*only super admin can create “admins” and “support agents”*

*\*\*admin can create only “managers”*

*// password min length 8 must use validation from frontend*

*// password includes : 1number, 1 uppercase, 1 lowercase, 1 special character*

Password encrypted by Md5

If user is not admin or super admin, he can not create new user

If user is chain admin, one can add multiple hotels

If user role is not chain admin or a manager, one can add only a hotel

Send new user data in response without password

***// get user by id***

**Get User By Id** **:** GET  /api/v1/user/get/:id

Required fields :

User Id *// in req.params.id*

*// Only super admin can see inactive or deleted user*

***// delete user by multiple ids***

**Delete User By Ids** **:** DELETE  /api/v1/user/delete/:ids

Required fields :

User Ids *// in req.params.ids*

//can select multiple ids for delete multiple users

Only super admin can delete user

*\*\*only super admin can delete user profile*

***// update user by id***

**Update User  By Id** **:** PUT  /api/v1/user/update/:id

Required fields :

           const { name, email, role, phone, profilePic, hotelId,

           isActive, isDeleted, supportAgentStatus, isChainAdmin

           } = *req*.body;

UserId // in req.params.id

Check if email is being updated and if it's already taken

If user is chain admin, one can add multiple hotels

If user is not chain admin or a manager, one can add only a hotel

Remove undefined fields

Update user

Send success response with update user data

*\*\*when user try to delete account it will not deleted but just update isDeleted field*

*\*\*only super admin can delete user profile*

***// Get all user***

**Get All User** **:** GET  /api/v1/user/getAll

Required fields :

const { search = "", role, isActive, hotelId, customerId

 } = *req*.query;

const page = parseInt(*req*.query.page) || 1;

const limit = parseInt(*req*.query.limit);

Find result length

Get result with pagination using “limit” and skip” without password field

*// search can be possible with multiple hotel ids and multiple customer ids*

*\*\* only super admin can see inactive users and deleted users*

Send success response with all user data

—-----------------------------------------------------------------------------------------------------------------------------------------

**Common routes :**

***// Temp File Upload***

**Create User** **:** POST  /api/v1/common/tempFileUpload

This is the route for store image/ file in temporary location on s3 server in temp folder

Used s3 configuration for store data

Will transfer file to folder later when every process will completed

Used “multer-s3” and “@aws-sdk/client-s3”

—-----------------------------------------------------------------------------------------------------------------------------------------

**Customer :**

***// Create a new customer***

**Create Customer:** POST  /api/v1/customer/create

const { name, email, password, phone, profilePic

 } = *req*.body;

*// password min length 8 must use validation from frontend*

*// password includes : 1number, 1 uppercase, 1 lowercase, 1 special character*

Password encrypted by Md5

Check if customer already exists

Create new customer with new data

Send new customer data in response without password

***// get customer by id***

**Get Customer By Id** **:** GET  /api/v1/customer/get/:id

Required fields :

Customer Id *// in req.params.id*

*// Only super admin can see inactive or deleted customers*

***// customer user by id***

**Delete Customer By Id** **:** DELETE  /api/v1/customer/delete/:id

Required fields :

Customer Id *// in req.params.id*

\**only super admin can delete customer profile*

***// update customer by id***

**Update Customer By Id** **:** PUT  /api/v1/customer/update/:id

Required fields :

           const { name, email, phone, profilePic, isActive,isDeleted

 } = *req*.body;

Customer Id // in req.params.id

Check if email is being updated and if it's already taken

*// only admission can update deleted or inactive user*

Remove undefined fields

Update Customer

Send success response with update Customer data

*\*\*when customer try to delete account it will not deleted but just update isDeleted field*

*\*\*only super admin can delete user profile*

***// Get all customers***

**Get All Customer :** GET  /api/v1/customer/getAll

Required fields :

const { search = "", hotelId, isActive } = *req*.query;

const page = parseInt(*req*.query.page) || 1;

const limit = parseInt(*req*.query.limit);

get all customers by multiple hotelId   // if hotelId exist

Find result length

Get result with pagination using “limit” and skip” without password field

Send success response with all customers data

*// search can be possible with multiple hotel ids and multiple customer ids*

*\*\* only super admin can see inactive users and deleted users*

***// Add to favorites and remove from favorites***

**Add Or Remove From Favorites  :** GET  /api/v1/customer/addToFavorites/:id

Required fields :

const { hotelId } = *req*.body;

const customerId = *req*.params.id;

Check in user favorites list if hotel is already in user favorites list,

 if exist remove hotel id and if not exist add hotel id to users favorite list

Count numbers of favorite hotel

Send success response with cout and hotel Id list

—-----------------------------------------------------------------------------------------------------------------------------------------

**Reviews :**

***// Create a new review***

**Create Review :** POST  /api/v1/review/create

const { customerId, hotelId, rating, comment } = *req*.body;

*// rating must between 1-5*

***// get review by id***

**Get Review By Id** **:** GET  /api/v1/review/get/:id

Required fields :

Hotel Id // in req.params.id

*// Only super admin can see inactive or deleted review*

***// delete reviews by ids***

**Delete Review By Ids** **:** DELETE  /api/v1/review/delete/:ids

Required fields :

Review Ids // in req.params.ids

// can select multiple ids for delete multiple reviews

Only super admin can delete reviews

*\*\*only super admin can delete hotel amenity*

***// update review by id***

**Update Review  By Id** **:** PUT  /api/v1/review/update/:id

Required fields :

Hotel Id // in req.params.id

  const { rating, comment, replyByHotel, isActive, isDeleted }    = *req*.body;

*// only admission can update deleted or inactive reviews*

Clear undefined values from payload

Update review

Send success res.

***// Get all reviews***

**Get All Reviews**  **:** GET  /api/v1/review/getAll

Required fields :

const { search = "", rating, startDate, endDate, hotelId,customerId } = *req*.query;

  const page = parseInt(*req*.query.page) || 1;

  const limit = parseInt(*req*.query.limit);

*// search can be possible with multiple hotel ids and multiple customer ids*

*\*\* only super admin can see inactive and deleted reviews*

—-----------------------------------------------------------------------------------------------------------------------------------------

**Master Room:**

***// Create a new master room***

**Create Master Room:** POST  /api/v1/masterRoom/create

const { hotelId, masterRoomName, subRooms, fullDayPrice, description, images, capacity, amenities, isActive } = *req*.body;

Check master room already exist in hotel with same name

Create master room

Create sub rooms with given array of “subRooms”

Upload Images of room

Send success response

***// get master room by id***

**Get Master Room By Id** **:** GET  /api/v1/masterRoom/get/:id

Required fields :

Hotel Id // in req.params.id

*// Only super admin can see inactive or deleted review*

***// delete master room by ids***

**Delete Master Room By Ids** **:** DELETE  /api/v1/masterRoom/delete/:ids

Required fields :

Review Ids // in req.params.ids

// can select multiple ids for delete multiple reviews

Only super admin can delete reviews

*\*\*only super admin can delete hotel amenity*

***// update master room by id***

**Update Master Room By Id** **:** PUT  /api/v1/masterRoom/update/:id

Required fields :

Hotel Id // in req.params.id

  const { rating, comment, replyByHotel, isActive, isDeleted }    = *req*.body;

*// only admission can update deleted or inactive reviews*

Clear undefined values from payload

Update review

Send success res.

***// Get all master room***

**Get All Master Room** **:** GET  /api/v1/masterRoom/getAll

Required fields :

const { search = "", rating, startDate, endDate, hotelId,customerId } = *req*.query;

  const page = parseInt(*req*.query.page) || 1;

  const limit = parseInt(*req*.query.limit);

*// search can be possible with multiple hotel ids and multiple customer ids*

*\*\* only super admin can see inactive and deleted reviews*

—-----------------------------------------------------------------------------------------------------------------------------------------