**SMS Portal Login Documentation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Edited By** | **Date** | **Reviewed By** | **Doc Version#** | **Comments** |
| Bhupesh Rai Choudhary | 27-03-2024 |  | 1.0 |  |

Repositry Details

Name: [JCT\_NJ\_LOGIN\_V2\_I](https://hcazdevops.ril.com/EA_Tech_Services/JIO%20Cloud%20Telephony/_git/JCT_NJ_LOGIN_V2_I?path=%2F&version=GBBhupesh.130125)

URL:https://hcazdevops.ril.com/EA\_Tech\_Services/JIO%20Cloud%20Telephony/\_git/JCT\_NJ\_LOGIN\_V2\_I

Branch: Bhupesh.130125

Name: [JCT\_NJ\_LOGIN\_S](https://hcazdevops.ril.com/EA_Tech_Services/JIO%20Cloud%20Telephony/_git/JCT_NJ_LOGIN_S?path=%2F&version=GBmaster)

<URL:https://hcazdevops.ril.com/EA_Tech_Services/JIO%20Cloud%20Telephony/_git/JCT_NJ_LOGIN_S>

Branch: lastProd

Name : [JCT\_NJ\_ConfigSetup](https://hcazdevops.ril.com/EA_Tech_Services/JIO%20Cloud%20Telephony/_git/JCT_NJ_ConfigSetup?path=%2F&version=GBmain)

URL: <https://hcazdevops.ril.com/EA_Tech_Services/JIO%20Cloud%20Telephony/_git/JCT_NJ_ConfigSetup>

**Objective :-**

This Documentation Highlights the Login Journey that is being used in the sms reports and sms reports admin portal along with the algorithms and screenshot and description and API curls

Following Scenarios will be covered in the documentation

* User login with 2FA ( with and without active session)
* User login with invalid password case
* User login with invalid OTP case
* Reset Password Journey
* Access Suspend Scenario - Custom Rate Limit via email
* OTP Rate Limit Exceeded
* User Do Not have access to reports portal

SMS Portal Url : <https://sms.jiocx.com/reports/auth/login>

SMS Reports Admin Url : <https://sms.jiocx.com/reportsAdmin/auth/login>

**Summary:**

**This section describes the login process along with the API used**

1. **/**validateEmail – email entered by user is validated and checked if its exist in DB collection ( SMSP\_USER\_MASTER : MD41) or not , if yes then they are asked for password
2. /genKey – on success of validateEmail , this API is called and it makes two key pairs one is stored in backend and other is sent in the response that is used by frontend for cipher text generation
3. /login - Once the user enters the password and proceed then password is sent in the form of cipher text and this cipher text is constructed with the help of key pair received IN genkey response and one time code that is generated on frontend side and this one time code is also sent to backend along with cipher text and once backend validates the password and user details , API success is indicated to frontend and OTP generation code is called , that contains logic to send OTP on phone and email
4. /verifyOTP – Once the OTP is submitted from verifyOTP screen , verify OTP is called and in this API first OTP is validated and once OTP validation check , active session is checked IN DB collection (login\_session\_history : MD40 )

* If active session exist then a very short token is set as cookie (approx 2-3 minutes ) along with other process
* Else no action

After checking active session then users details are fetched from (SMSP\_user\_Master details – MD41 & SMSP\_USER\_DOMAIN\_MAPPING – MD42) collection

In case This details will be kept in JWT token that will be used as cookie and also act as a redis key while validating the cookie and the value of this redis key (JWT TOKEN) are the user details

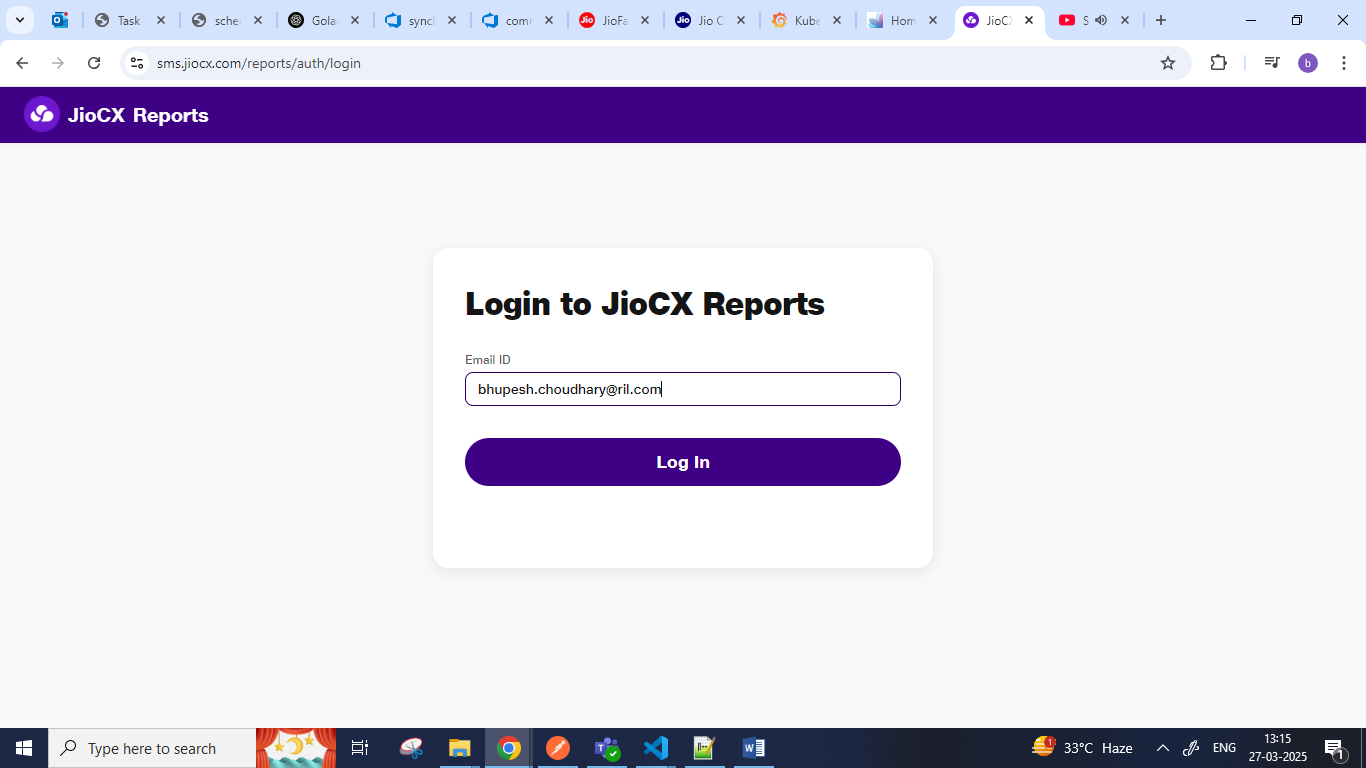
From user details only , some flags are checked if user logged in first time or user password has expired as they will be sent in the final response

1. If active session doesn’t exist then token is set as cookie and session history is saved in DB collection (login\_session\_history : MD40 )
2. If active session exist then only final response is returned along with very short token used as a cookie

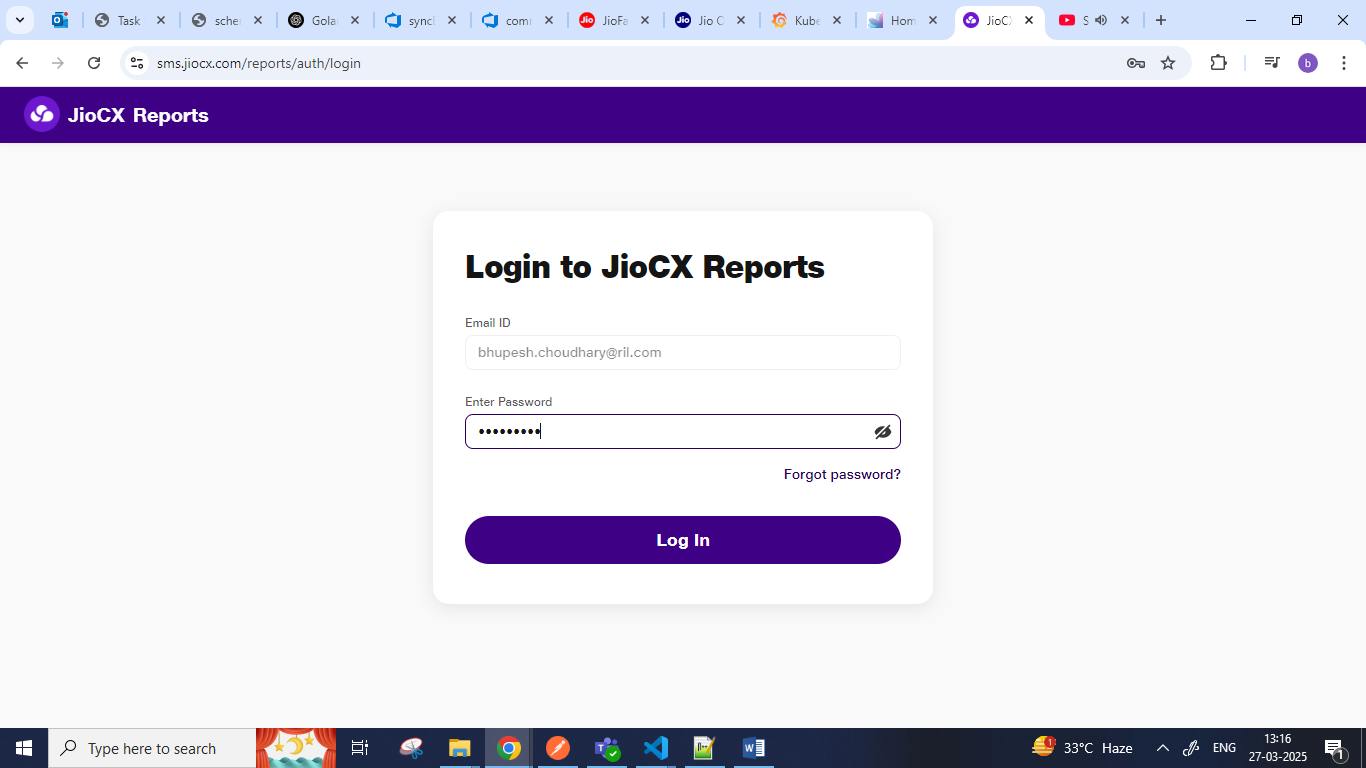
**Scenario 1: Successful User login with correct password and OTP**

**Screen Journey:**

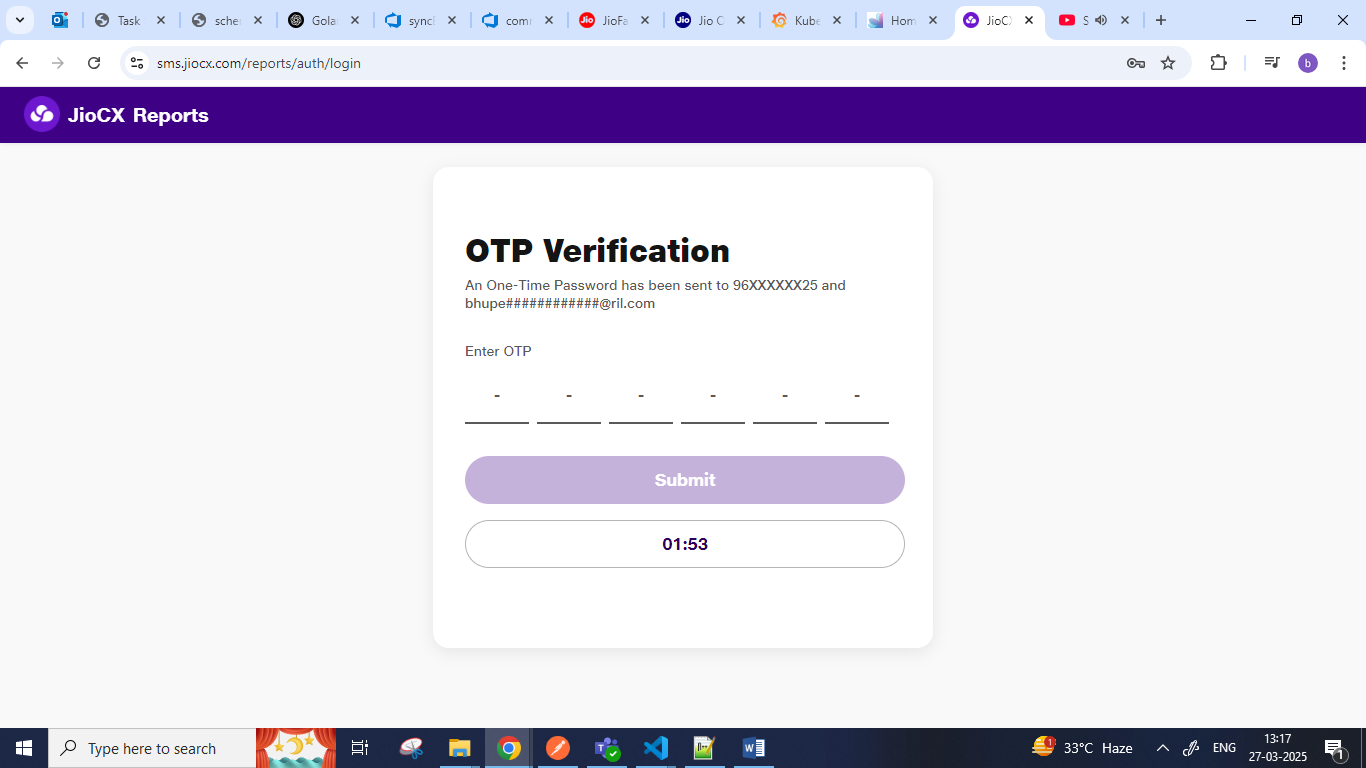
Step 1: User enter correct email



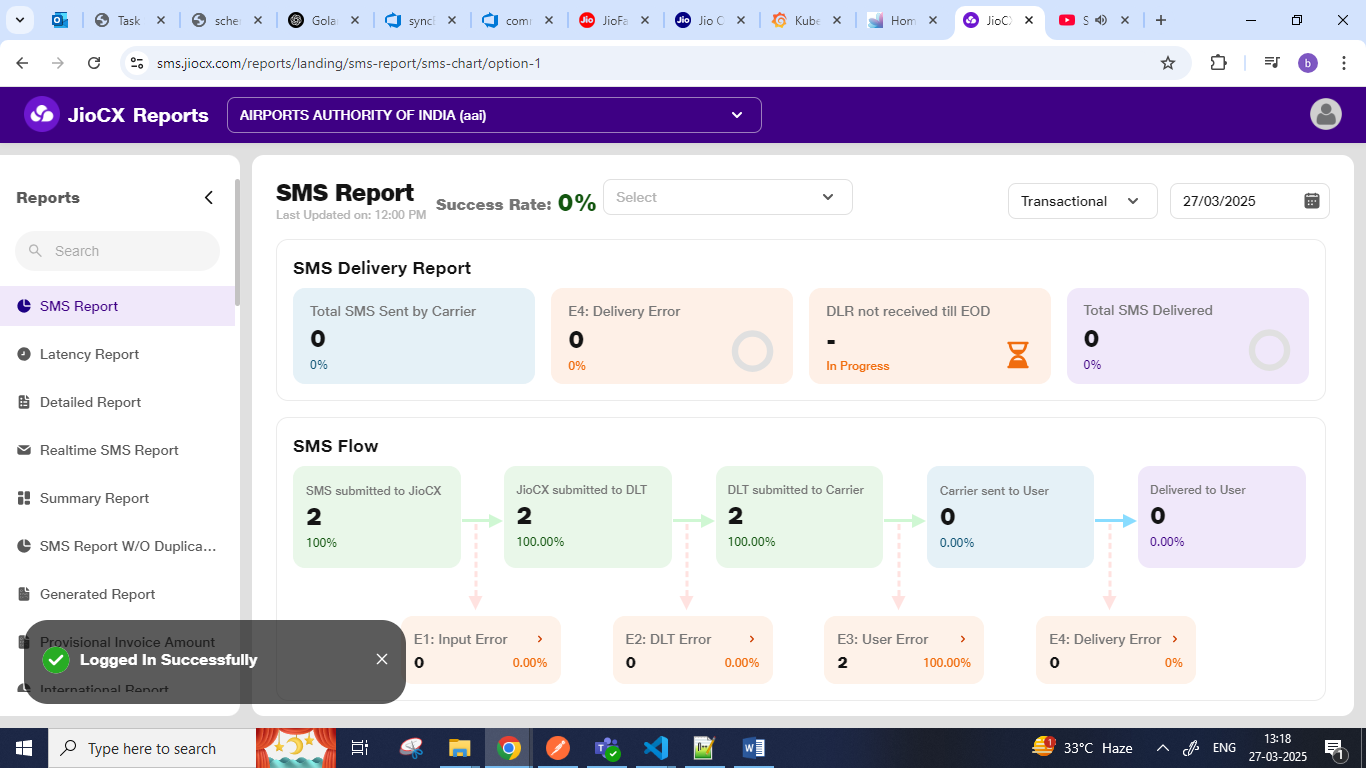
Step 2: If Valid User they need to enter their password



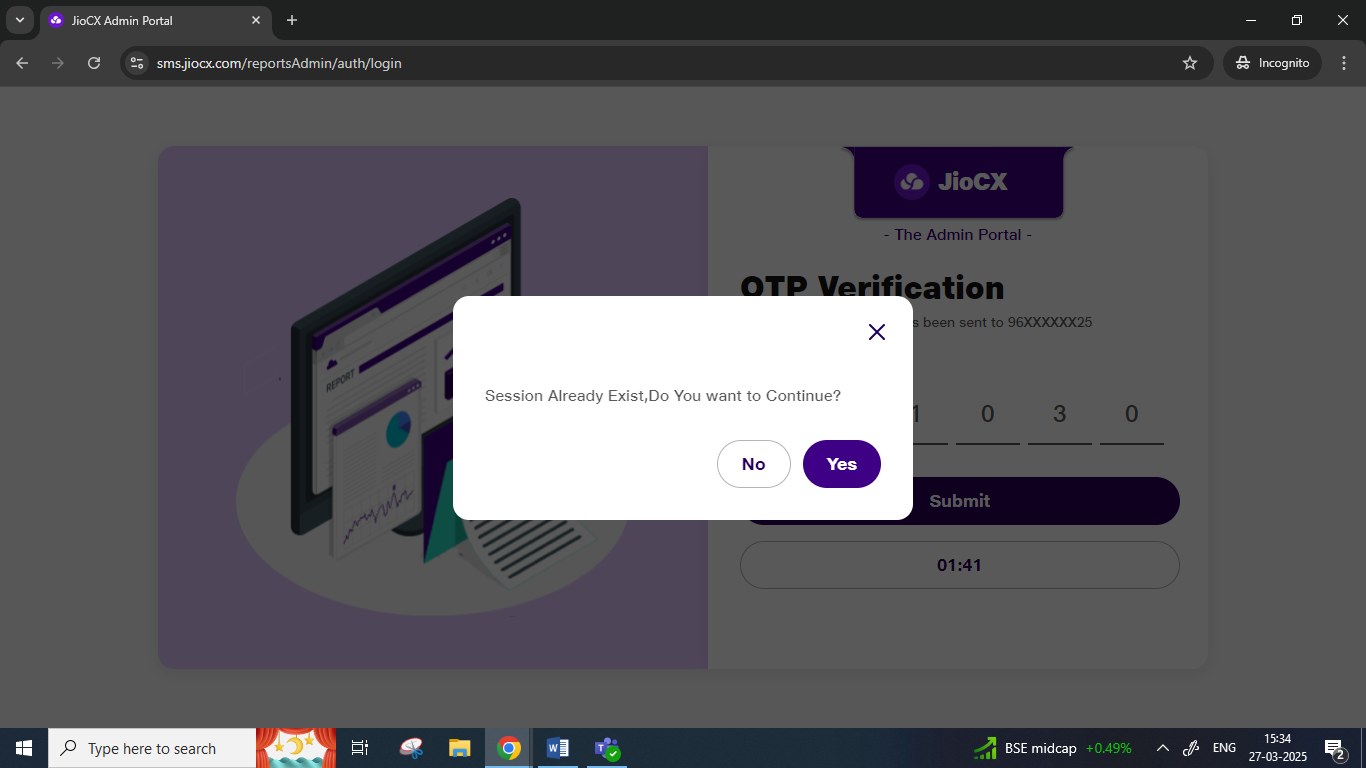
Step 3: User Entering Correct password will get screen to enter OTP



Step 4: After Entering the OTP received on mail or mobile, user is logged into portal:



Case 2: Successful user login with active session already there ,in that case user gets an option to either override the existing session or cancel the current login attempt

  
  
In the above screen on clicking NO, user is routed back to login screen

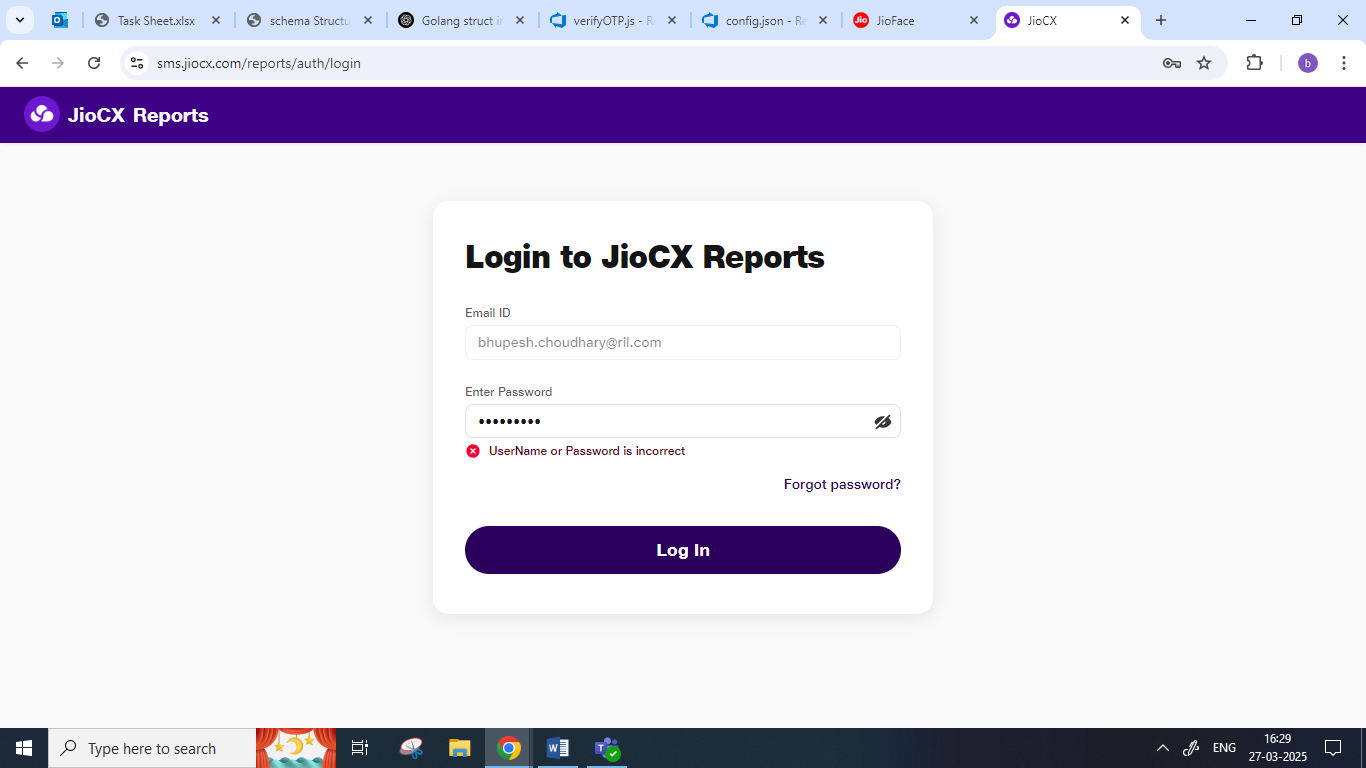
On clicking yes, user session is override and new session becomes active and old session cookie are discarded

API Used with Respective Sequence:

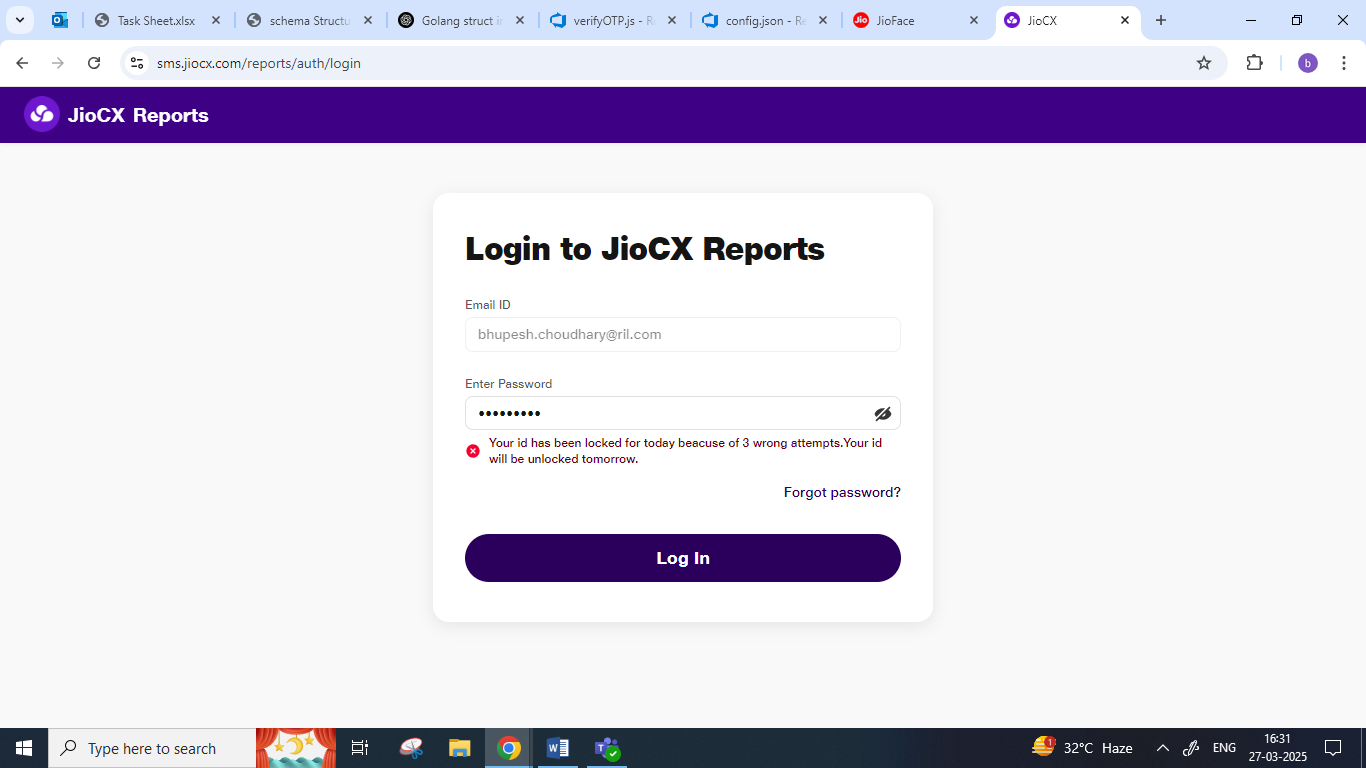
1. /validateEmail – This API check if user is present in DB or not
2. /genKey – This API is used to generate public and Secret key pairs that is sent to frontend in the response and frontend will use that to send password as a cipher text to backend in the login API
3. /Login – This API Authenticates the User based on password provided along with sending OTP on mobile and email
4. /verifyOTP – If Password Authentication is successful , then to verify the OTP sent to user , verifyOTP is called , it will verify the OTP for user along with setting cookie for that session, that will be used to authenticate the user on each request
5. In Case of Case 2, /overRideUserSession API is called if user choose to proceed on existence of a session

**Scenario 2: User login with Invalid Password:**

**Once user submits the password and if the password is wrong then they have 2 more attempts to provide correct password, after that there account is locked and super admin needs to unlock their account and they will receive your account is locked message on attempting login again**

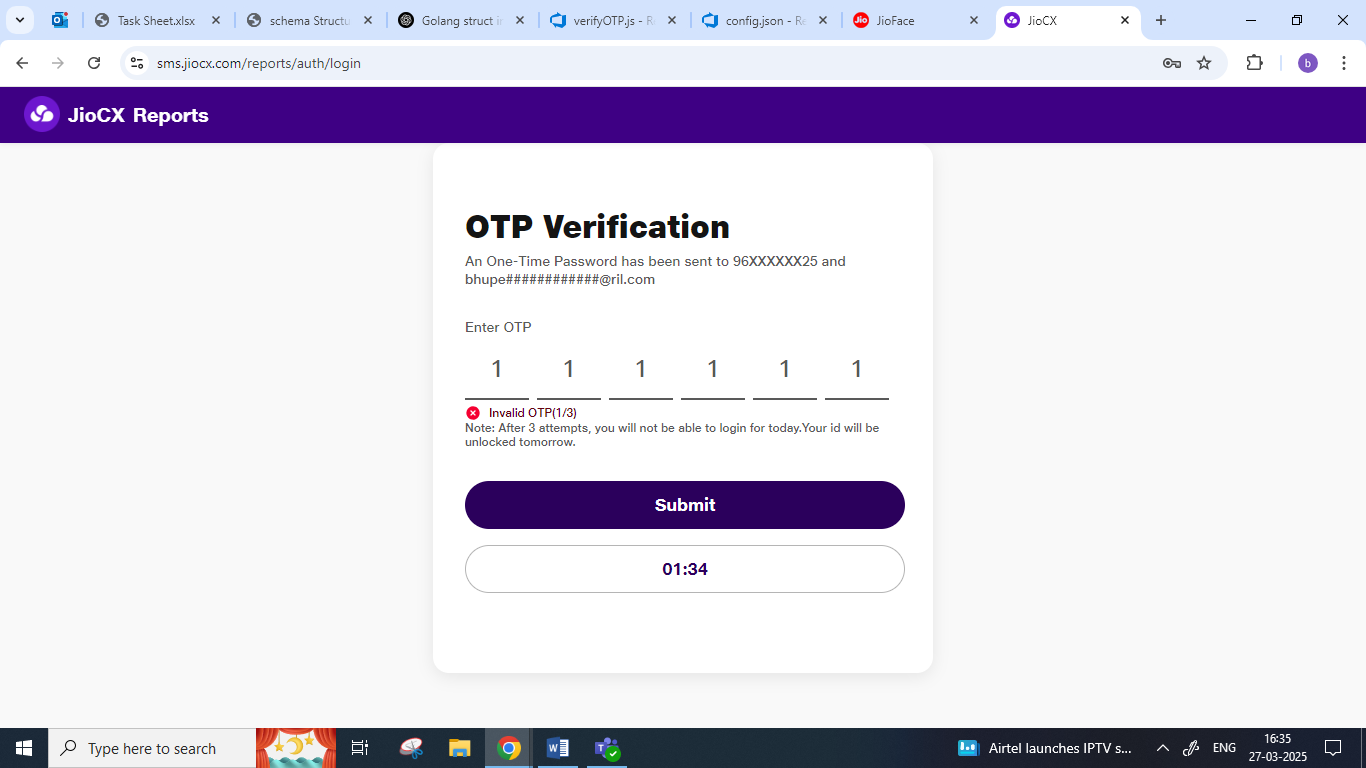


**After three wrong attempts user won’t be able to login and their account needs to be unlocked**

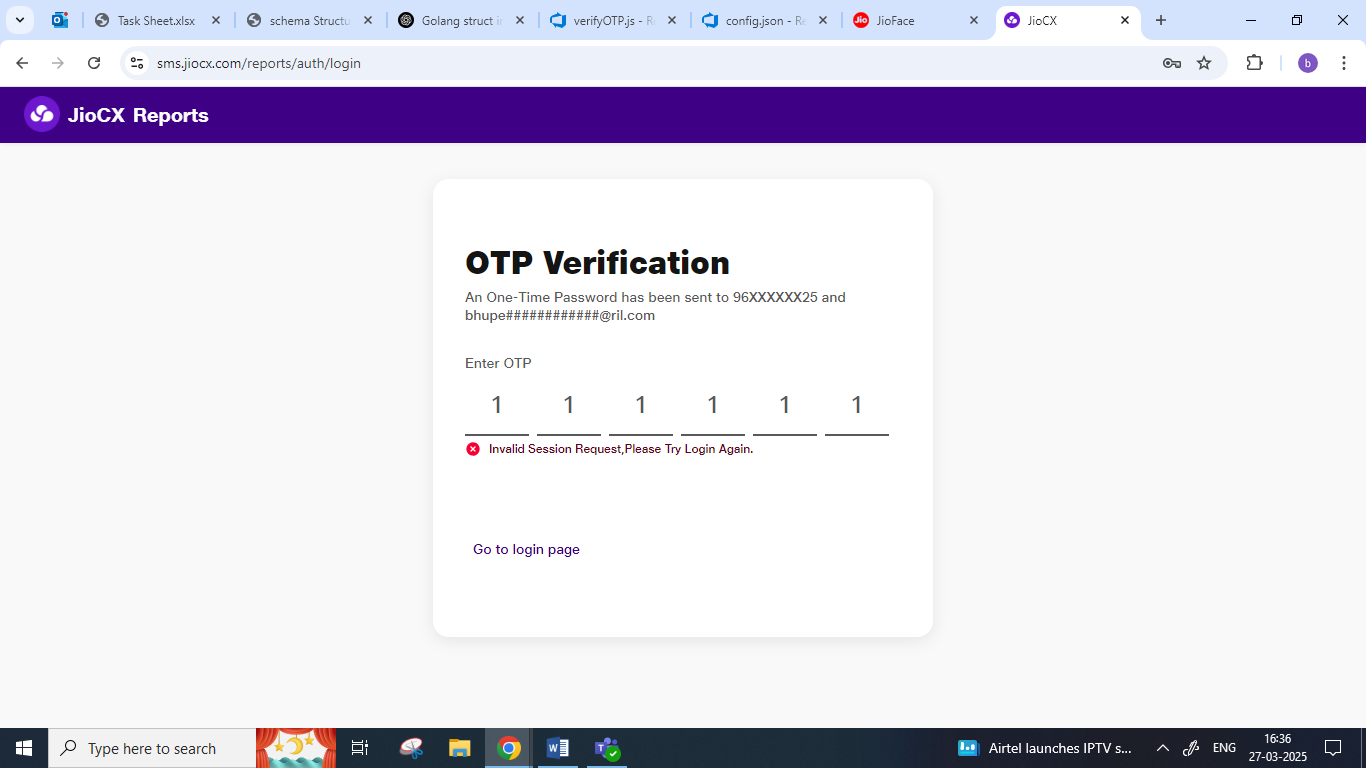


**Scenario 3: User login with Invalid OTP:**

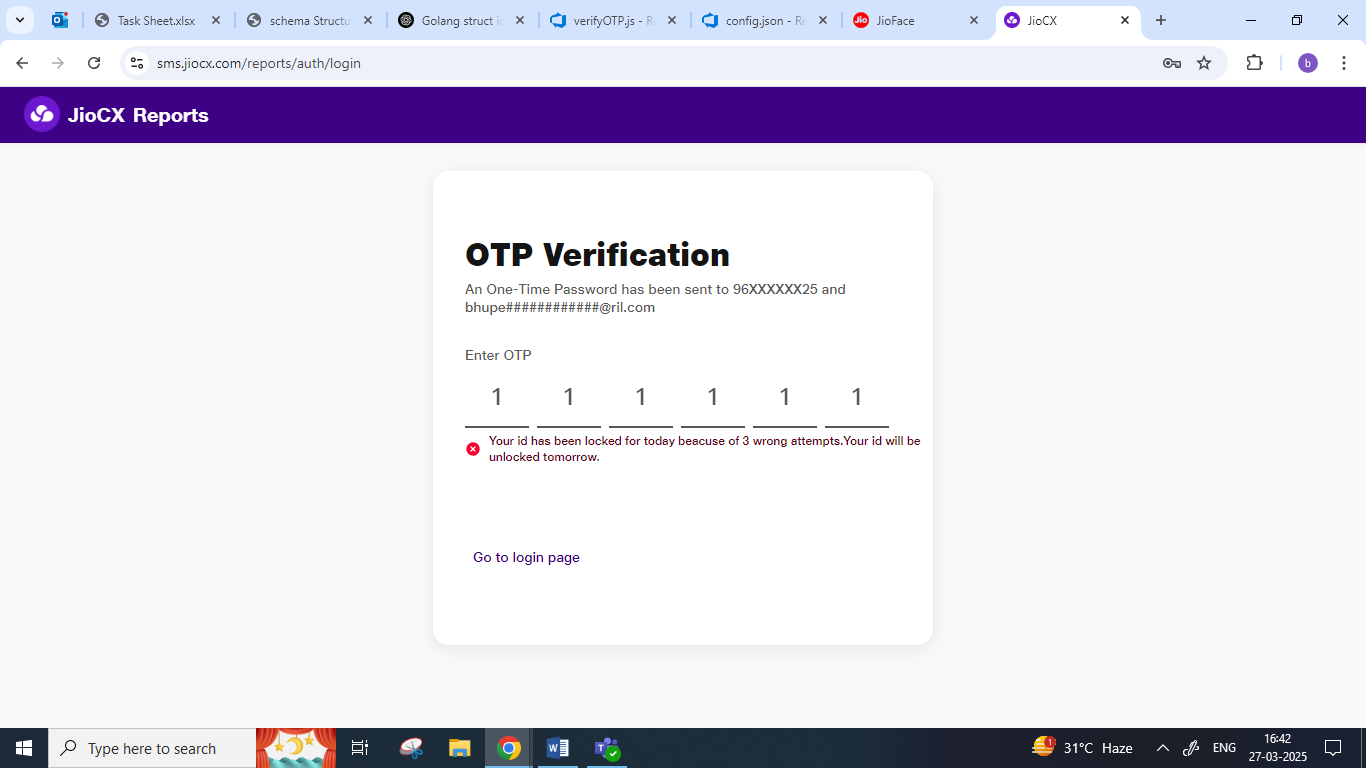
**On Password authentication, if user fails the OTP check 3 times then also their account will be locked and super admin needs to unlock their account**



**If User fails the current OTP attempt then they need to re login by providing email and password then providing correct OTP else below message will be there**



**On 3 Wrong attempts account will be locked and below screen will appear**

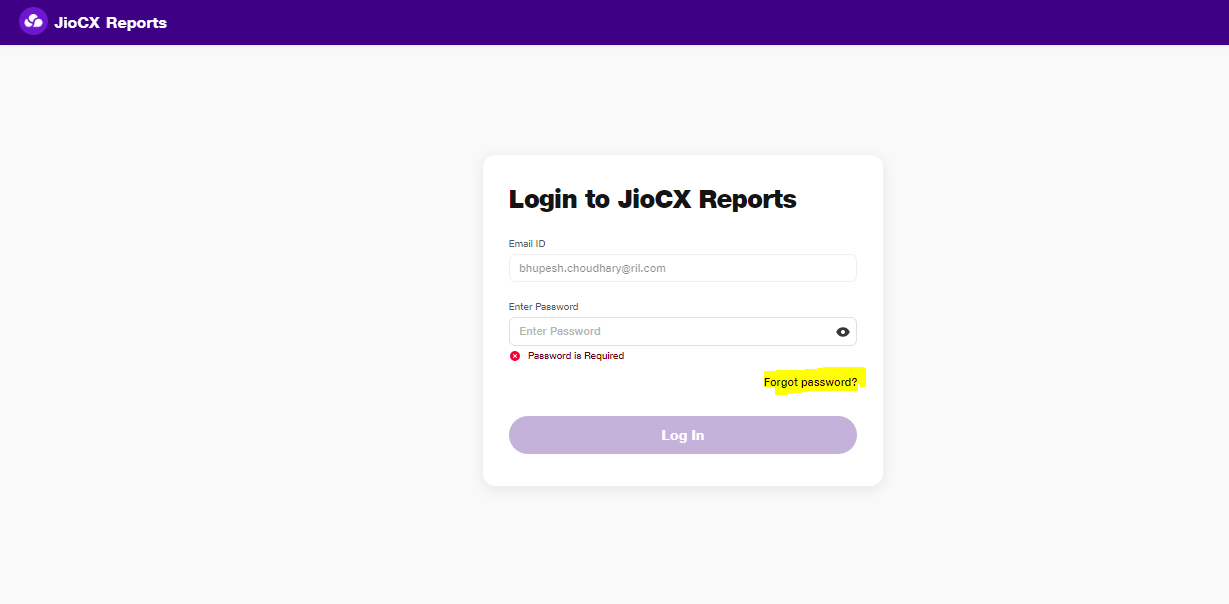


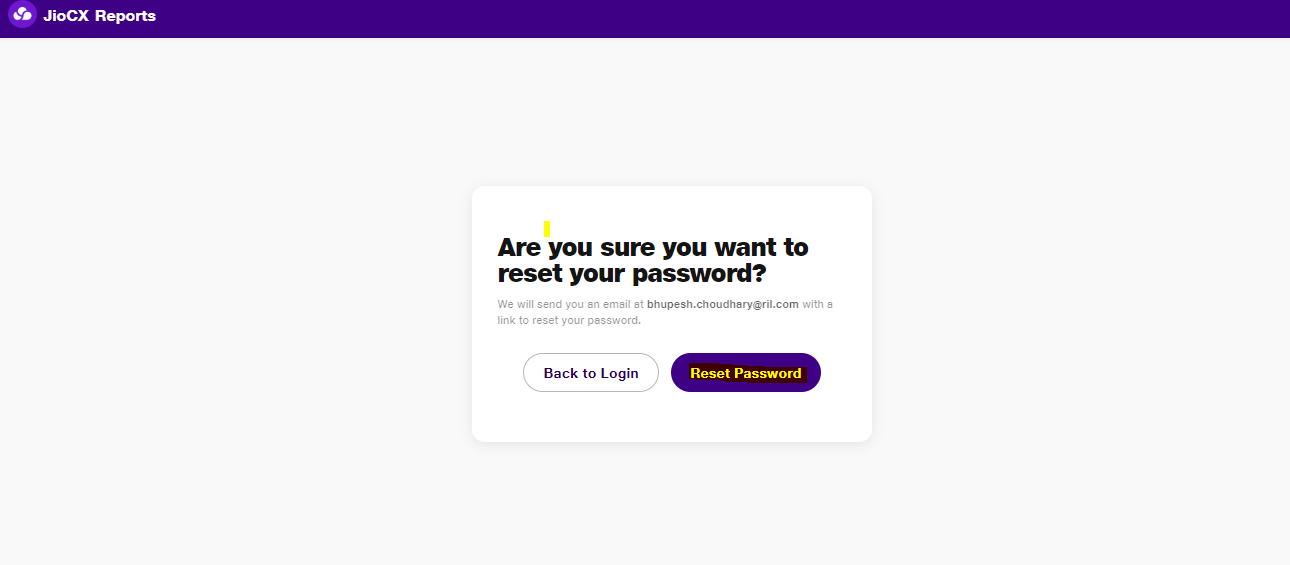
Scenario 4: Reset Password Journey

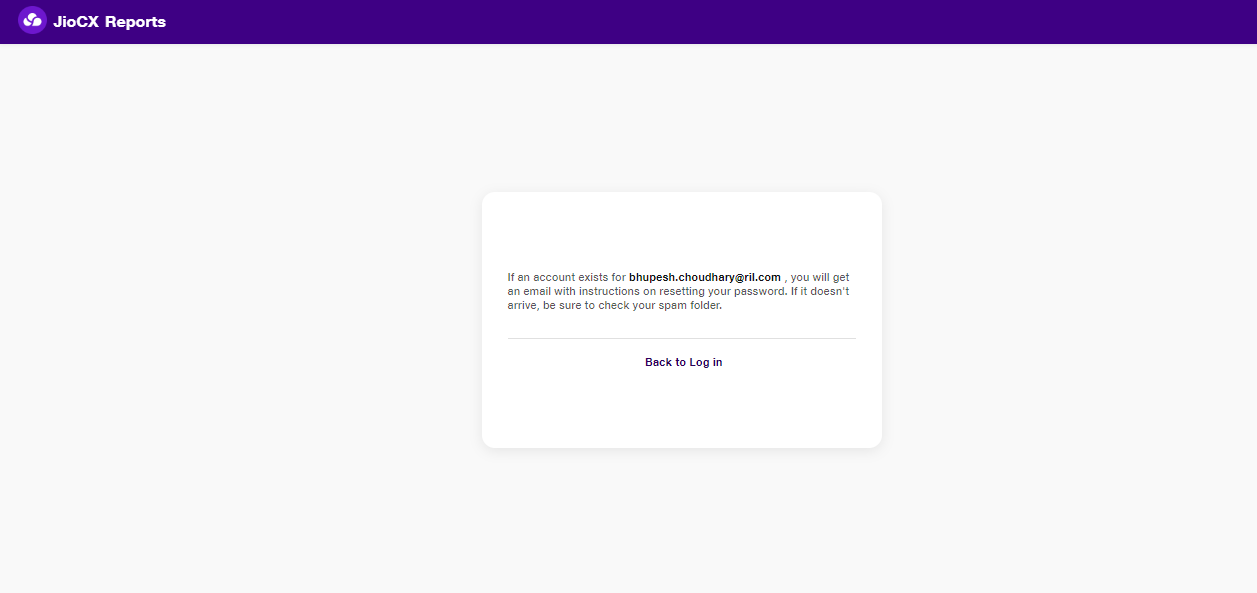
User’s can reset their password by providing valid mail Id and clicking on forgot password button

After clicking on forgot password then need to confirm they want to go back to login or reset password

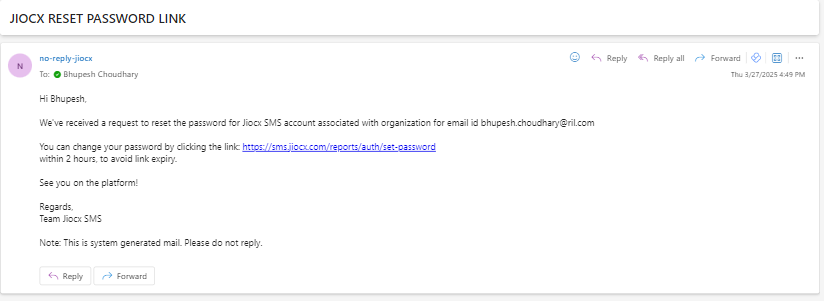
If they choose reset password then confirmation popup is shown and reset password is link is sent over mail



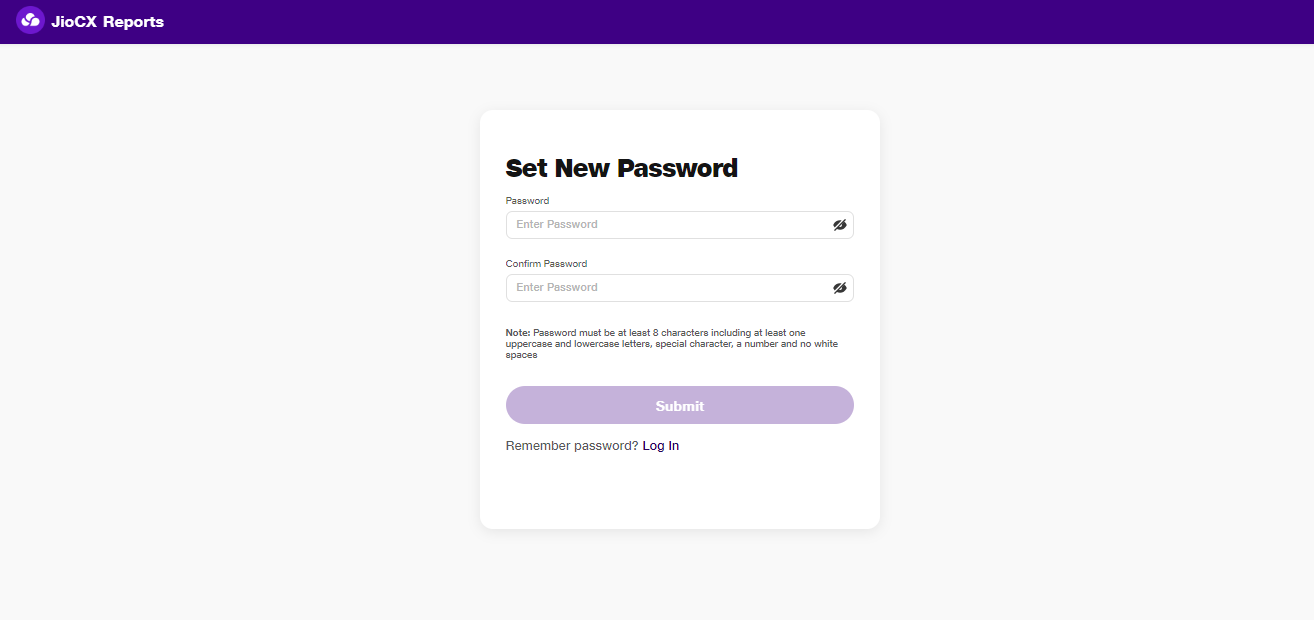




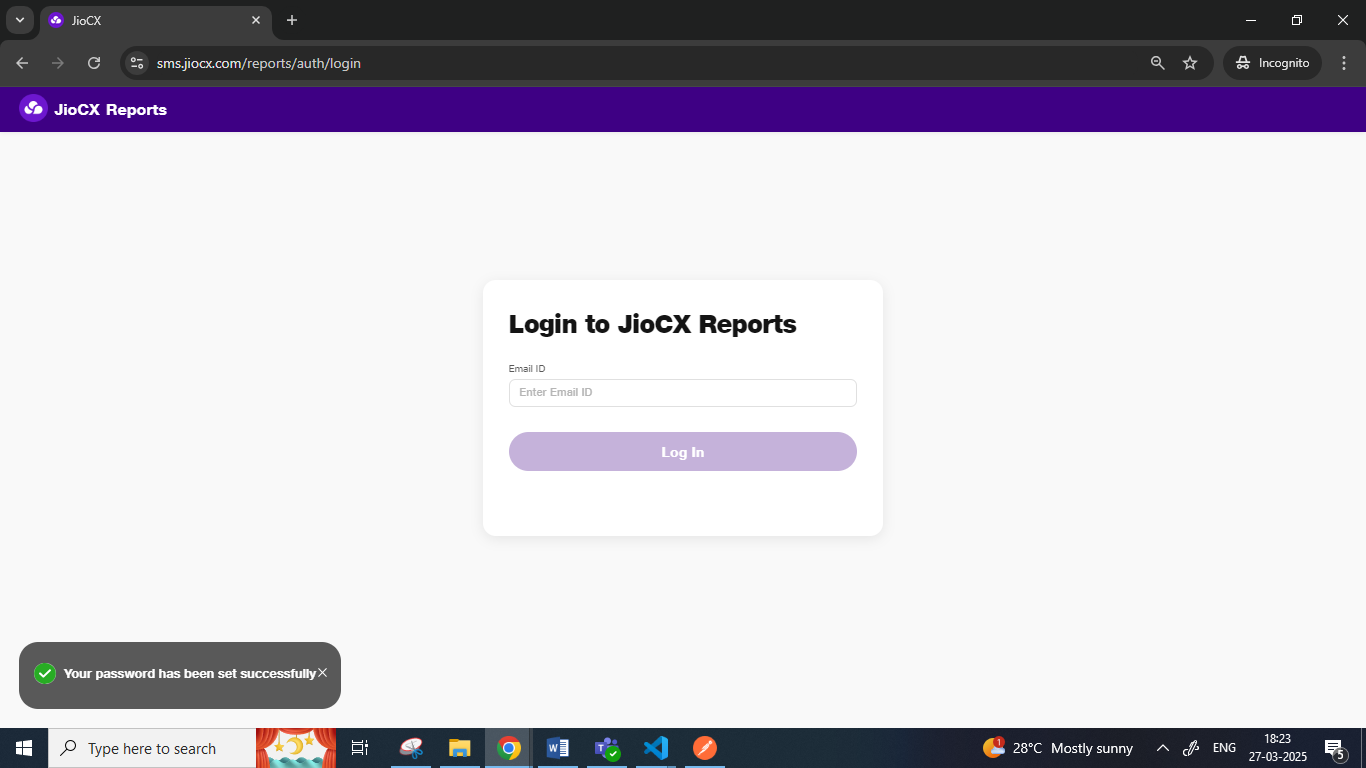
Below screen shows the mail view in which reset password link is received on user side

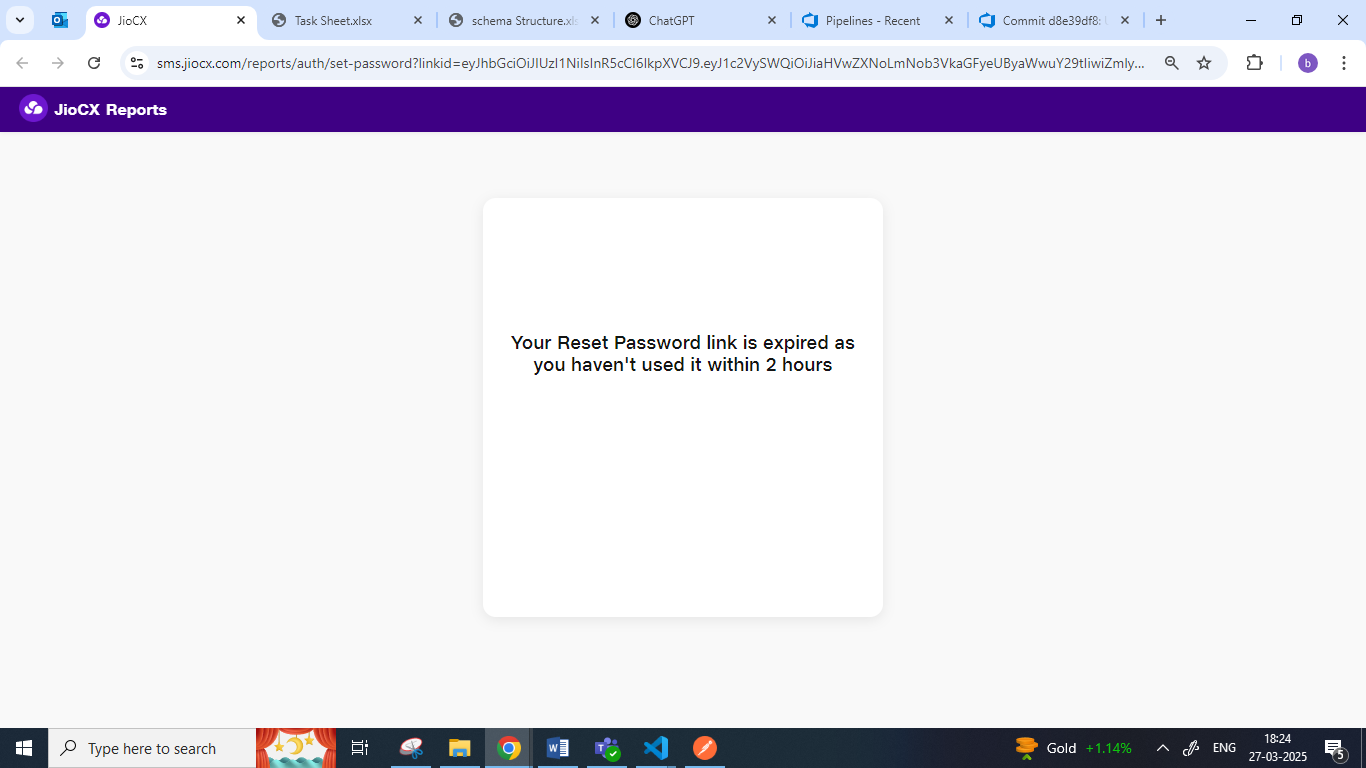


On accessing the link, it will redirect to below screen where user can reset their password:

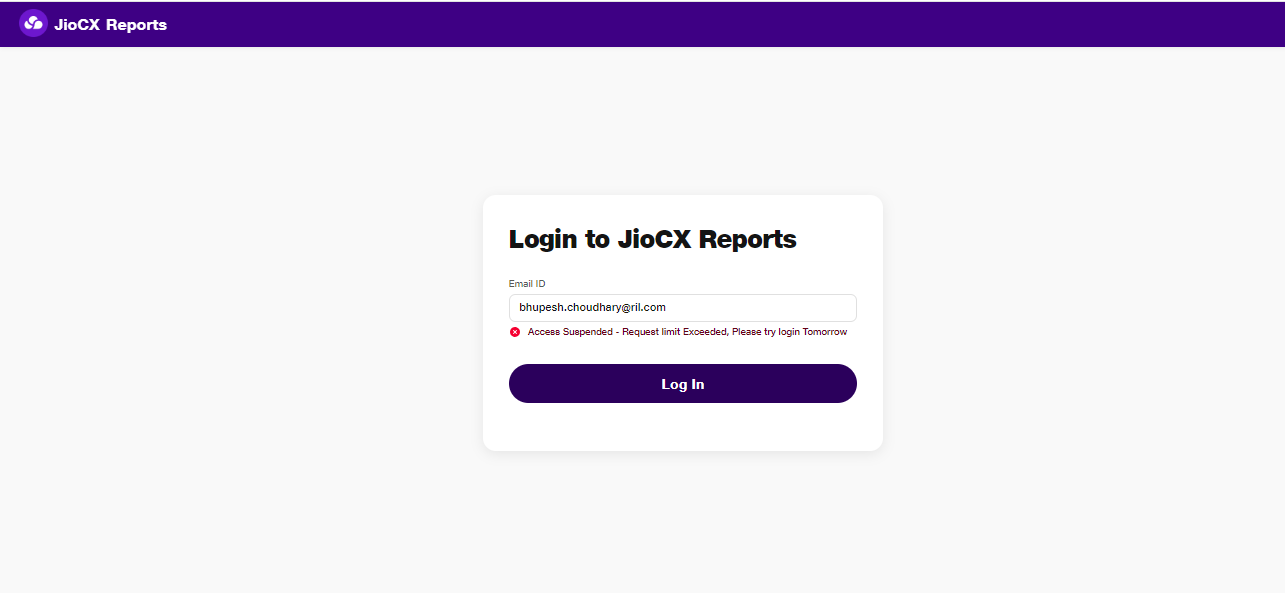


On successful reset , below screen will show :



In Case of link expiry following screen will show :   
  


**Scenario 5: Access Suspended for a user because they have tried login more than 15 times**



This limit will reset on the next day , in case this needs to be manually reset for a user then use below curl :

curl --location 'https://sms.jiocx.com/api/jct-home-s/setInRedis' \

--header 'Content-Type: application/json' \

--data-raw '{

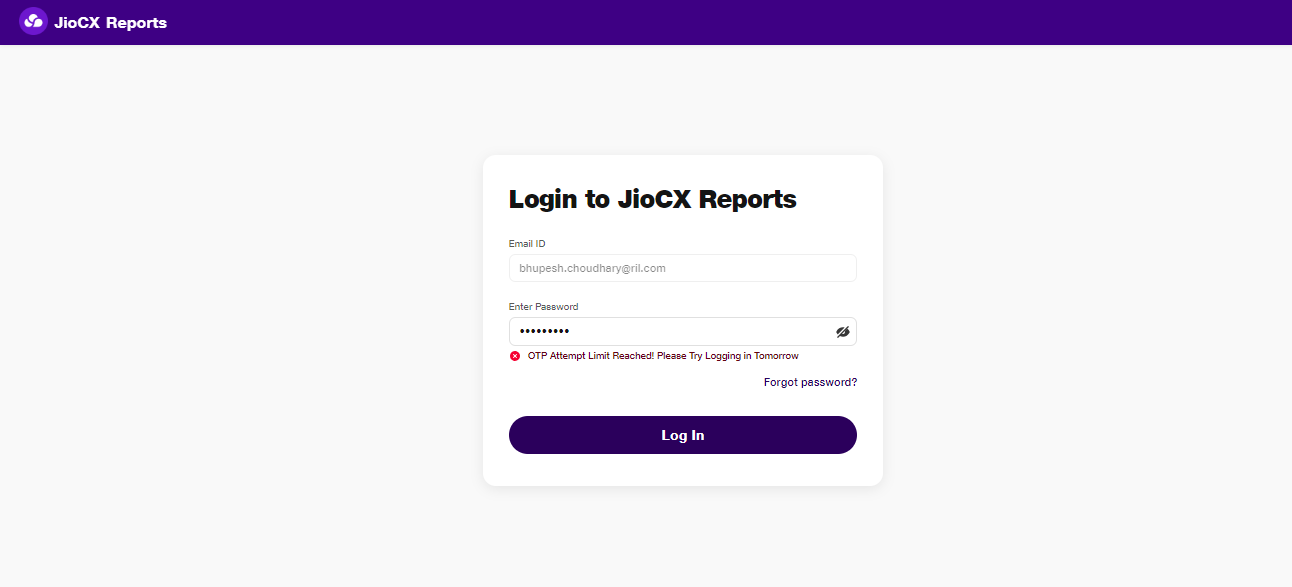
"redisKey": "userEmail/request/rateLimit",

"data": "",

"redisExpireTime": 0

}'

**Scenario 6: OTP Rate Limit Exceeded for a day if user attempts to send OTP 10 times in a day**



This limit will reset on the next day, in case this needs to be manually reset for a user then use below curl:

curl --location 'https://sms.jiocx.com/api/jct-home-s/setInRedis' \

--header 'Content-Type: application/json' \

--data-raw '{

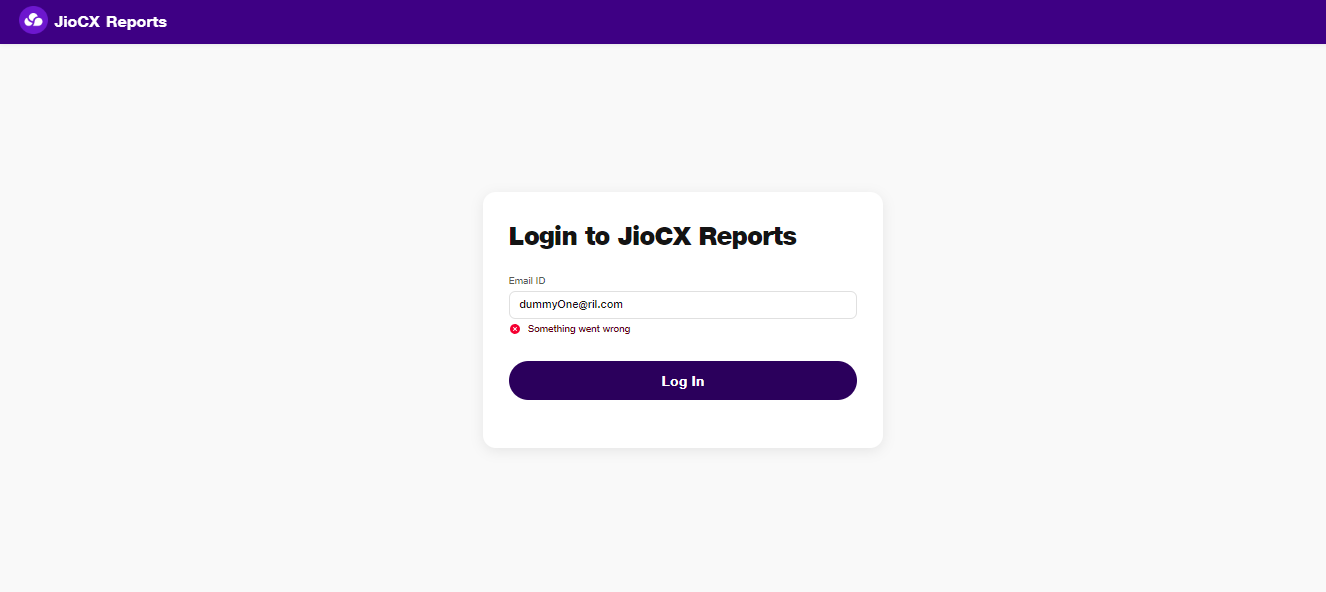
"redisKey": "bhupesh.choudhary@ril.com/loginv2/otpRateLimit",

"data": "",

"redisExpireTime": 0

}'

**Scenario 7: User Don’t have access to reports portal**



Something went wrong message will appear

**API Algorithm Explained:**

1. ValidateEmail : Used to Validate if User is a valid user or not

Algorithm:

1. Check if Input Email is Valid or Not
2. Check in redis if user is locked or not
3. Query DB collection ( SMSP\_USER\_MASTER : MD41) and if data is found then
4. Check if they are active , if not return "Email ID is inactive"
5. If they are active and they are not in unlock state , "Email ID is locked"
6. If they are active and unlock then return status true with domain map count
7. If they are not found in DB return “Something went Wrong” Response
8. genKey – Used to Generate public key and secret key pair , that will be used by frontend to supply password as cipher text in the /login API

Algorithm:

1. Validate Input Email , if Its Valid Proceed else return Validation Error
2. Generate two key pairs with the help of nacl.box.keyPair() method

This method is a part of below package

const nacl = require("tweetnacl");

nacl.util = require("tweetnacl-util");

1. In the key pair there will be a public key and secret key so store the public key from first key pair and secret key from second keypair in redis parallel to rediskey: `JCT/${emailId.toLowerCase()}/KEY`
2. Take the Publick Key from second key pair and secret key from first key pair and return it as a response
3. login – Used for user authentication and OTP send and generate

Algorithm:

1. take username , password in form or cipher text and onetime code as input
2. validate inputs if invalid return error else proceed
3. check if user account is locked , if yes then return locked msg else proceed
4. query mongodb collection (SMSP\_user\_master : MD41) to check if user with given username is present and also if its active , unlocked and its userType is “Customer”, “Aggregator” and “Entity”
5. If user found then decode its password and verify hash with password hash stored in DB
6. If password is wrong then maintain invalid password attempt count parallel to redis key : `${username}/loginv2/passwordAttempt`
7. After 3 wrong attempts lock the user account in DB by setting userState to locked and in redis by setting the rediskey `${username}/loginv2/accLock`, the redis lock is valid till EOD , User if locked in DB can be unlocked by Super Admin Only
8. If password is correct then call genOTP Code that will generate a OTP and send it to mobile and Email when Flag is “b”
9. Also Maintain a key in redis -  `${username}/loginv2/otpRateLimit`- this will count the otp attempts by a user
10. Return success response with masked email and mobile number on which OTP’s was sent
11. VerifyOTP – this API check if input OTP is same as the OTP that was sent to user and provides authorization or allow user to login by setting a cookie and override a previous session

Algorithm

1. Check redis key (`${email}/loginv2/passwordAttempt`) data , if not present then it’s an invalid session probably some calling verifyOTP one after another instead of try login again
2. If its valid session then reset this key and check OTP input with OTP set in redis key `${email}/jct/signup/${otpflag}`
3. If OTP not match then maintain invalid OTP count in redis parallel to redis key `${email}/signupv2/otpattempt` and return the attempt count with incorrect response
4. After 3rd wrong attempt along with setting the attempt count lock the user in db as well as in redis , the redis lock is valid till EOD
5. If Input and redis OTP matches then proceed to check if a session exist of user or not
6. From collection (login\_session\_history : MD40) check session history or not , if not exist then return false else return true along with setting a very short token in redis for approx. ( 2-5 minutes)
7. This session existence response is further used in the code
8. If session already existed then user data is returned along with activeSession flag as true
9. If session was not there then a user data is queried and JWT token is made using some user data fields and refresh token is also made and session history is saved in DB and final response is sent to user and in that refresh token is present in response body and token is set as cookieValue for cookieName = “JCTSSO”
10. In the JWT token ( Cookie ) following information is present

{

    userId: “email Id of user”,

    type: “wheather user is Aggregator, Entity or Customer”,

    organisationName: “org name of customer that user is a part of”,

    domain: “domain of customer, aggregator or entity”,

    accessToken: true, ( indicates it’s a cookie or short token )

    isAdmin: “Values will be PU, BU, AG OR EN”, user is a admin update flag to “PU” else if user is aggregator update flag to “AG” else for entity user flag is “EN”

    activeSession: true or false // if user already has a session it will be true else it will be false

};

1. In Refresh Token ( Long Token) following information is present

{

    userId: “email Id of user”,

    type: “wheather user is Aggregator, Entity or Customer”,

    organisationName: “org name of customer that user is a part of”,

    domain: “domain of customer, aggregator or entity”,

 refreshToken: true, ( indicates it’s a refresh or long token )

    isAdmin: “Values will be PU, BU, AG OR EN”, user is a admin update flag to “PU” else if user is aggregator update flag to “AG” else for entity user flag is “EN”

    deleteFlag: true or false // based on user data , if user is deleted or non deleted

  };

1. In the final response , based on user data some other parameters also sent like code
   1. 5001 – if user is logging into account first time, they will be routed to password change screen
   2. 5002 – user password was last changed 60 days before they need to reset it
   3. 5003 – normal case
   4. activeSession – if its true then user session already exist they will get popup if they need to override previous session or they want to cancel the current login attempt
   5. refreshToken – on first login ,frontend will get this value that will be stored in localStorage, in case a session already exist then refersh token will be sent as empty

Based on above values in verifyOTP response, frontend will route user to the screens

1. /getUserList – this API is used to get user details specifically used by frontend to determine if user belongs to aggregator , entity , normal user or an admin user along with domain’s they have access to

Algorithm:

1. Fetch userId from cookie
2. If no userId return invalid response
3. Based on userId from cookie fetch user data from DB , if its active user
4. Also fetch data from (smsp\_user\_master:MD41) along with (smsp\_reports\_master –MD44) and (smsp\_user\_reports\_mapping:MD45)
5. After data is fetched from the user master data check if user is logging in first time or its password has expired , if yes then assign code 5001 or 5002 else 5003
6. Then to find reports that should be shown to user pass the userData , reports master data and data from user reports mapping to reportsDataFormat
7. In reportsDataFormat function , if user is admin then do noting as admin can see all reports available and its controlled on fronted , if user is not admin thenn
8. Create a map from reports master data , by using key field as map key and its result as obj
9. Check if any report array is present is in user reports mapping entry if yes then set isUserMappingSpecified to true and make a separate map for user mapped reports
10. If no report array present or no user mapping entry, fetch data from (smsp\_customer\_master:MD43), assign the reports assigned to customer to reportsData
11. If reportsData has length greater then 0 then
12. Now in a loop on reports data check each entry and if user mapping specified flag is true then check userspecificreports map else check reportsMap for all reports
13. Make an object for the looped entry and push in a array and then based on sequence number sort the entries in final report array
14. Return the sorted result , this is the array that describes the report that should be visible to user
15. After getting the result in menuResult variable
16. Make payload to get data from MD42 and MD43 where category is “Customer” , newArchitecture variable is true and delete\_flag is false
17. In case user is a customer add then customer code also in the fetch payload of MD43 as we need to fetch data for only that customer
18. In Case user is not admin but belongs to category “Aggregator” or “Entity”, fetch data for that aggregator or entity that the user belongs to and in its response we will get all customers that are part of that aggregator (this customer array will contain id’s of customer present in MD43)
19. Update the domain payload to fetch entries for customer that has Id in customer array of aggregator or entity ( if userType is one of them)
20. Fetch the data based on final domainPayload
21. Take a flag and assign it “BU”
22. IF user is a admin update flag to “PU” else if user is aggregator update flag to “AG” else for entity user flag is “EN”
23. Return the final response with user details its domain array and flag along with reports that should be visible to them and some flag like downloadOTP , unmaskMobilenumber etc are decided based on usermaster entries else that will be decided based on customer master entry if multiple customer array there then it will be based on 1st customer values , that is flag set for first customer in aggregator or entity list will apply to all the customer present in that aggregator or entity
24. overRideUserSession – In case of active session of employee , it will help to override the old session and create a new session

Algorithm:

1. from the cookie , get the userId
2. fetch session data for that user from MD40 collection
3. based on the fetched data , keep the accessToken and refreshToken value in variable and delete that session entry using the \_id for the record
4. remove the accessToken , refreshToken from redis as well
5. create new session and refersh token same as verify otp case where user session data is not present and final response will be same as verifyOTP -session not exist case
6. logout – delete the current session and remove its data

Algorithm:

1. from cookie fetch the userId
2. fetch session data from MD40 based on userId
3. delete the session and remove the session data from redis as well based on token obtained from cookie
4. return the success response
5. refreshToken – this api is used to renew a session in case a user’s access token expires but refresh token is valid , this API will be called by frontend when status code 418 is sent by backend API

Algorithm:

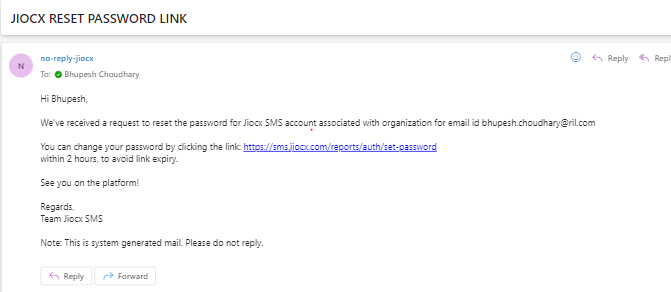
1. take refresh token from input
2. check if refersh token is valid by fetching redis data for the token
3. if not valid return else
4. decode the refesh token and make a token object with details and query login session based on email present in decoded token
5. compare the input and refreshToken from DB , if match then create new access token based on token obj and update the token in login session
6. return both the refresh token and access token as response
7. sendFPLink – use to send mail to reset password along with a reset link

Algorithm:

1. take email as input and validate it
2. after validation check if reset password attempt was made previously or not
3. check rate limit based on username passed as-`${req.body.emailId}/sendFPLinkV2`, passed to rateLimiting function , if rate limit exceeded return error else
4. fetch userData data based on email and generate a token based on user data and set in redis also maintain redisKey- `${req.body.emailId}/resetPasswordAttempt` with token as value for period same as link validity period
5. send the mail for reset password where hyperlink will contain the linkId with token whereas while display no token is displayed

exampleLink:

<https://sms.jiocx.com/reports/auth/set-password?linkid=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJiaHVwZXNoLmNob3VkaGFyeUByaWwuY29tIiwiZmlyc3ROYW1lIjoiQmh1cGVzaCIsImxhc3ROYW1lIjoiQ2hvdWRoYXJ5IiwiaWF0IjoxNzQzNDA1OTQ0LCJleHAiOjE3NDM0MzQ3NDR9.2hZtdzsWIAvypj8eQXJSjDtDF0UrwmgV8uXIKKrViHI>



1. ValidateFPLink – This is used to validate the reset password link if its valid or not , if valid then only user will be allowed to reset the password

Algorithm:

1. In the headers , we will get a resetId , this resetId sent by frontend is same as the token that was sent as value to linkId
2. Check if the token is present in redis if yes then link is valid and proceed
3. Clear any JCTSSO cookie present in browser through res.clearcookie
4. Set JCTSSSO cookie In the browser that will be used or passed to reset password code
5. Return success
6. saveResetPassword – this is used to reset the password in case of forgot password after reset link is validated

NOTE : Before this API genKey is called and its output is used by frontend to send new password value along with a oneTimeCode

Algorithm:

1. take the input as newpassword and onetime code as input and userId from the cookie
2. from the cookie in headers take the JCTSSSO cookie check , check the token if its present in redis or not
3. if not present return password link expired message else
4. take the ontimecode and newpassword value that is in cipher text form decode it and generate its hash
5. update the password hash for the user along with passwordChangeOn date
6. after the password is reset , delete any JCTSSO or JCTSSSO token from browser and delete the redis value for token as well
7. return success
8. setNewPasswordOnExpiry – this API is used to set password on expiration of password

Algorithm: same as saveResetPassword

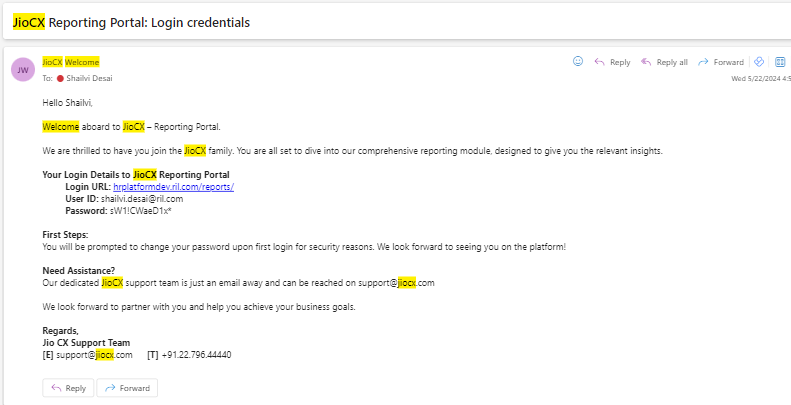
1. sendOTP – this API is used to sendOTP on mobile and email, in case of resend OTP scenarios where user didn’t entered otp within 2 minutes

Algorithm:

1. take input and validate
2. check if user account is locked or resend otp attempt has been exceeded by user
3. if not count the current attempt and generate the OTP
4. store the OTP in redis and send on email and mobile
5. registerUserReports – This API is used to add user for reports and billing portal from reports admin portal

Algorithm:

1. in input is isBilling flag is true then
2. user need to be saved in collection ( jct\_admin : MD01) as it belongs to billing portal
3. validate the input
4. generate the hash value for password received from input
5. check if user already exist or not based on mobile number or email
6. if not then make the save payaload for MD01 and MD02 and save the data in DB
7. no email will be sent to user in this case
8. if is billing was false then validate the input
9. auto generate a password along with its hash
10. fetch the details of customer (domain) for which the user is being added
11. check if user exist or not , if already exist return error
12. insert data in user collection along with domain mapping collection
13. send the mail to user being added along with link to reports portal , user name and auto generated password , reference mail attached



Collection Details:

|  |  |  |
| --- | --- | --- |
| MDCODE | COLLECTION NAME | Remarks |
| MD01 | jct\_admin | Contains details of billing portal user with their name , email , password |
| MD02 | admin\_domain\_mapping | Contains domain mapping entry for each individual user |
| MD40 | login\_session\_history | On user login, the session and refreshtoken is saved in this collection to check for or override active session when user starts a new session |
| MD41 | smsp\_user\_master | Conatins details of reports portal user along with details , userType and flags like getDownload otp, unmask mobile numbers etc |
| MD42 | smsp\_user\_domain\_mapping | Contains domain mapping entry for each individual user |
| MD43 | smsp\_customer\_master | Customer weather aggregator, entity or customer are added along with their data and also it contains reports assigned to that customer and user that belongs to that customer and other details related to invoicing and billing |
| MD44 | smsp\_reports\_master | It’s a master for all the reports that is available to assign to different customer , all reports needed to be present in reports master before assigning to customers |
| MD45 | smsp\_user\_reports\_mapping | While assigning reports , if reports assigned to a user and no at customer level so that particular user will see reports assigned to user level instead of customer level |
| MD46 | smsp\_billing\_confirmation | Billing confirmation entries |
| MD47 | smsp\_aggregate\_access\_mgmt | In case the customer type is entity or aggregator then which all customers has been assigned or are a part of that aggregator or entity is present in this collection inside customer array field  This customer array will conatin \_id’s of customer ( domain) that are a part of that aggregator or entity  This data will be used in getUserList when sending domain list that should be visible to user based on entity or aggregator the user is a part of |

Limits and Config Values

* To send OTP on mobile below api is being used :

URL : <https://jiocxint.jiocx.com/apggw/jiocxint/sms/v1/send>

* To send OTP on email or other mails as well below API is being used :

Url: <https://jiocx.jiocampaigns.com/interface/v2/swift/create-swift-batch>

API-Key - eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjdXN0b21lcl9pZCI6OTAsImlhdCI6MTcwNzQ2MzE0N30.GbrNYMmcTs6ey17fXY79Qp6ievCnRECLMpqfMm0sSgY

POC: [Abhishek3.Ranjan@ril.com](mailto:Abhishek3.Ranjan@ril.com) / [Manish9.Rai@ril.com](mailto:Manish9.Rai@ril.com)

* In a day , User can try sening OTP – 10 times, exceeding this will result in otp limit exceeded error
* In a day , user can log in to their account -15 times, exceeding this will result in access suspended message
* Access token that is used as cookie is valid for – 5 hours
* Refresh token that is valid for – 8 hours
* Very short token valid for – 2 min
* To enable or disable default OTP in any environment , generateDefaultOtp flag inside OTP object needs to be configured
* OTP length configured is – 6 digits
* useOldOtpMechanism – this flag is used to shift OTP send mechanism to old mechanism

Url: <http://jct-nj-sms-s.jiocloud.svc.cluster.local:9327/api/jct-sms-s/redis/lpush>

* for testing user , testingEmails array is being maintained , seprate OTP need to be set for them as they will not receive the OTP on phone or mail

curl --location '[https://sms.jiocx.com/api/jct-login-s/setInRedis'](https://sms.jiocx.com/api/jct-login-s/setInRedis%27) \  
--header 'Content-Type: application/json' \  
--header 'Cookie: BIGipServerHRPF-DEV-POOL-31380=rd40o00000000000000000000ffff0a1a18f2o31380; BIGipServerHRPF-DEV-POOL-31380=rd40o00000000000000000000ffff0a1a1968o31380' \  
--data-raw '{  
    "redisExpireTime": 2000000000000000000000,  
    "redisKey":  "krushna.fuke@ril.com/jct/signup/m",  
    "data": 123450  
}'

* for server based rate limit , in 5 minutes 100 request are allowed on a particular pod – this is implemented using the “express-rate-limit” node package

 "serverRateLimit": {

    "timeinMinutes": 5,

    "requestAllowed": 100

  }

* "userType": ["Customer", "Aggregator", "Entity"] – only user with these types are currently allowed to login to reports and reports admin portal and "userState": "unlock" should be there
* SMS and Email related values are maintaining in config for the repo used