Client management tool

Table of contents

- **Introduction**
- **Tech Stack Used**
- **Installation & Setup**
- **Project Structure**
- **Authentication & Authorization**
- **API Endpoints (Backend)**
- **Database Schema & Models**
- **Architecture Overview**
- **Deployment**

Introduction

- **Project Name**: Client Management Tool
- **Author(s)**: Bikash Kakati, from Truelift Team.
- **Version**: v1.0.0
- **Link**: https://user-management.develop.truelift.ai/
- **Description:** The **Client Onboarding Tool** streamlines the user and client management process within **Truelift** organization. It provides a structured workflow where **Admins** oversee the onboarding process, including **Truelift user registration, Approvals for database creation, and permission management**, ensuring a secure and organized system. Authorized **Truelift Users** handle day-to-day operations such as **client onboarding, user management, and email notifications**, while all key actions require **Admin approval**, maintaining control and compliance. The system also maintains a

detailed approval history for future reference, ensuring transparency and accountability in all onboarding activities.

Tech Stack Used

Layer	Technology Used
Backend	Node.js, Express.js, MongoDB, Mongoose, Cookie-parser, Cors, JsonWebToken, Nodemailer, Bcryptjs
Frontend	React.js, TailwindCSS, Shadcn, Redux-Toolkit, React-router-dom, Lucide-react, React-toastify,
Deployment	Aws Lambda, Apigateway, Aws Amplify,

Installation & Setup for local environment

- Node.js v18+
- MongoDB (Local or Atlas)
- Git Bash

Backend Setup

```
git clone https://github.com/TrueLift/Truelift_dev_backend.git
cd Truelift_dev_backend
npm install
```

- Set up .env in the root and add these followings

```
SERVER_PORT =

ALLOWED_CLIENT =
MONGODB_URL =

ADMIN_PASSWORD =

# JWT
TOKEN_KEY =
TOKEN_NAME =



# MAIL
SMTP_PASS =
SMTP_USER =
SMTP_PORT =
SMTP_HOST  =
```

- Finally

```
npm run dev
```

Frontend Setup

```
git clone https://github.com/TrueLift/Truelift_dev.git
cd Truelift_dev
npm install
```

- Set up .env in the root and add these followings

```
VITE_BASE_URL = http://localhost:3030/api/v1
```

- Finally

```
npm run dev
```

Project Structure

Backend

```
backend/
 │── controllers/      # Business logic
 │── models/          # MongoDB Schemas
 │── routes/          # API Routes
 │── middlewares/      # Verify JWT, Custom Response
 │── utils/           # Helper functions
 │── views/           # Tamplates(Email)
│── .github/         # CI/CD Script for auto deployment
 │── index.js         # Main entry
 │── server.js        # All Initializers
```

Frontend

```
frontned/
 │── src/
 │   ├── components/   # Reusable UI Components
 │   ├── pages/       # React Pages
 │   ├── store/       # Redux store, react query slices
 │   ├── utils/       # Utility functions
 │   ├── constant/     # Some constant variables
 │   ├── App.js       # Main App component
 │   ├── index.css     # All tailwind and utility css
 │   ├── index.js      # Started of react
```

Authentication & Authorization

Roles

- **Truelift Admin**: All permissions- Truelift user registration, Approvals, Send onboarding completion emails to client users.
- **Truelift User**: Create client organizations, Create database, Create users and send for approvals

Terminologies used in the documentation

- **Truelift Admin**: Admin who have permission to register trulift users and approvals
- **Truelift User**: User who is member of **Truelift** organization and has permission to Create client organizations, Create database, Create users and send for approvals
- **Client**: Onboarded organizations
- **User**: Users inside every client, who will going to use our **Truelift** platform.

API Endpoints (Backend)

- Some of the Methods are not following best practices because api gateway for cookie not accepted those methods like - Delete, Put etc
- Endpoints Start with **/api/v1** then this endpoints

Common endpoints

Method	Endpoint	Description
POST	/login	Login user (Truelift User/Truelift Admin)

Method	Endpoint	Description
POST	/logout	Logout user (Truelift User/Truelift Admin)
POST	/change-password	To change the password of login
GET	/user-details	To get logged in user details (name, role, email etc)
GET	/clients	Get all onboarded client list
GET	/users/:clientId	Get all users registerd inside an client
GET	/user-tracking/:clientId/:userId	Get all tracking details for an user inside
GET	/database-details/:clientId	Database details for client
GET	/database-trackings/:clientId	Database tracking details

Truelift admin only

Method	Endpoint	Description
POST	/accounts	Create Truelift user account
POST	/approve-database	Approve the database creation request
POST	/approve-user	Approve the user creation request
POST	/send-email	Send mail to the users which approved

Truelift user only

Method	Endpoint	Description
POST	/create-client	Create New Client
POST	/create-database	Send Approval Request to the admin for Database creation

Method	Endpoint	Description
POST	/create-user/:clientId	Send Approval Request to the admin for User creation
POST	/edit-user/:clientId/:userId	Send Approval Request for user details updation
POST	/delete-user/:clientId/:userId	Send Approval Request for user details deletion
GET	/get-collections	Get Collection options which we want to include at the time of Database creation for Client

Database Schema & Models

Truelift User Schema

- **DB Collection Name**: Truelift_users
- **Purpose**: To store the users (**Truelift Admin, Truelift Users**) which are allowed to use the Truelift client management tool.
- **Description**:
    - Password will store in hash format using **bcrypt**

```
const usersSchema =  new Schema({
    name:{
        type: String,
        require:[true, "Name is required"],
    },
    email: {
        type: String,
        unique: [true, "Email is already exist"],
        require: [true, "Email is required"],
    },
    role:{
       type: String,
       enum: ["admin","user"],
       default:"user",
       require: [true, "role is required"],
    },
    password:{
       type: String,
       require: [true, "Password is required"],
       select: false,
    },
}, {timestamps:true})

export const TrueliftUsers = model("trueliftUsers",usersSchema,"Truelift_users")
```

Temp Client Schema

- **DB Collection Name**: Temp_clients
- **Purpose**: To store clients details, users inside the client and also database details for temporary purpose until Approved.
- **Description**:
    - The Client id is connected with Approved database schema
    - User id for users connected with allowed users schema.

```
const tempClientSchema = new Schema({
    clientName:{
       type: String,
       require:[true, "Client name is required"],
    },
    createdBy:{
       name:{
          type: String,
          require:[true, "Name is required"],
        },
        email: {
            type: String,
            require: [true, "Email is required"],
        },
        createdDate: {
            type: Date,
            default: Date.now,
            require: [true, "created date is required"]
        }
    },
    users:{
        type: [
          {
            email: {
                type: String,
            },
            role:{
                type: String,
                enum: ["admin","user"],
                default:"user",
            },
            emailSendStatus:{
                type: Boolean,
                default: false,
            },
            emailSendDate:{
                type: Date,
                default: Date.now(),
            }
          }
        ]
    },
    database:{
        databaseName: {
            type:String,
            default:"",
        },
        collections:{
            type:[
                {
                    type:String,
                }
            ]
        },
        approvedStatus:{
            type: String,
            enum: ["pending","accepted", "rejected"],
            default:"pending",
        }
    }
}, {timestamps:true})

export const TempClient = model("temporaryClients", tempClientSchema,"Temp_
```

Dynamic Schemas

- **DB Collection Name**: Not a Schema
- **Purpose**: To store all schemas that will show to **Truelift user** at the time of Dynamic database creation and including database collections for specific clients.
- **Description**:
    - We can Include as many schemas as we want and all create dynamically within the database

```
export const dynamicSchemas = {
  GL_Business_tags : {
    model:"GL_Business_tags_schema",
    schema: GL_Business_tags_var
  },

  GL_Dataset_array: {                // collection name
   model:"GL_Dataset_array_schema", // model name
   schema: GL_Dataset_array_var     // schema name
  },

  GL_Dataset_details: {
   model:"GL_Dataset_details_schema",
   schema:GL_Dataset_details_var
  },

  GL_Dataset_raw: {
   model:"GL_Dataset_raw_schema",
   schema: GL_Dataset_raw_var
  },

  GL_Dataset_result: {
   model: "GL_Dataset_result_schema",
   schema: GL_Dataset_result_var
  },

  GL_Experiment_details: {
   model:"GL_Experiment_details_schema",
   schema: GL_Experiment_details_var
  }
 };
```

Approval Schema

- **DB Collection Name**: Approvals
- **Purpose**: To store all type approval request sent to **Truelift admin** and change status of those approvals. This also helps to show the tracking ui for user and database.

- **Description**:
    - It exist in two main types - Database and User approval.
    - And also has subtypes that requests are different action types - Create, Edit and Delete .
    - Comment in the approval request only given by admin when it is rejected.

```
const approvalsSchema = new Schema({
    clientId:{
        type: mongoose.Schema.Types.ObjectId,
        ref:"temporaryClients",
    },
    userId:{
        type:mongoose.Schema.Types.ObjectId,
    },
    type:{
        type:String,
        enum:["database", "user"],
        default:"user",
    },
    subType:{
        type:String,
        enum:["edit","delete","create"],
        default:"create"
    },
    createdBy:{
        name:{
            type: String,
            require:[true, "Name is required"],
        },
        email: {
            type: String,
            require: [true, "Email is required"],
        },
        createdDate: {
            type: Date,
            default: Date.now,
            require: [true, "created date is required"]
        }
    },
    verifiedBy:{
        name:{
            type: String,
            default:"",
        },
        email: {
            type: String,
            default:"",
        },
        verifiedDate: {
            type: Date,
            default: Date.now,
        }
    },
    status:{
        type: String,
        enum: ["pending","accepted", "rejected"],
        default:"pending",
        require: [true, "status is required"],
    },
    comment:{
        type: String,
    },
})

export const Approvals = model("approvals",approvalsSchema,"Approvals");
```

Approved database Schema

- **DB Collection Name**: Approved_Database
- **Purpose**: To store the approved database and collection
- **Description**:

```
const approvedDatabase = new Schema({
    clientId:{
        type:Schema.Types.ObjectId,
        ref:"tempClients",
        require:[true, "Client id is required"],
    },

    databaseName:{
        type:String,
        require:[true, "Database name is required"],
    },

    collections:{
        type:[
            {
                type:String,
            }
        ],
        require:[true, "Collection names are required"],
    },

})

export const ApprovedDatabase = model("approvedDatabase", approvedDataba
```

Allowed user Schema

- **DB Collection Name**: Allowed_users

- **Purpose**: To store those users which are approved by **Truelift Admin**, these users now can able to sign up in the **Truelift Platform**.
- **Description**:
    - **org** basically help us to dynamically connect with that database, for what user allowed to use.
    - S3 field also will come which help users to fetch and upload file in a specific S3 bucket(Not had that feature till now in the **Client management tool**

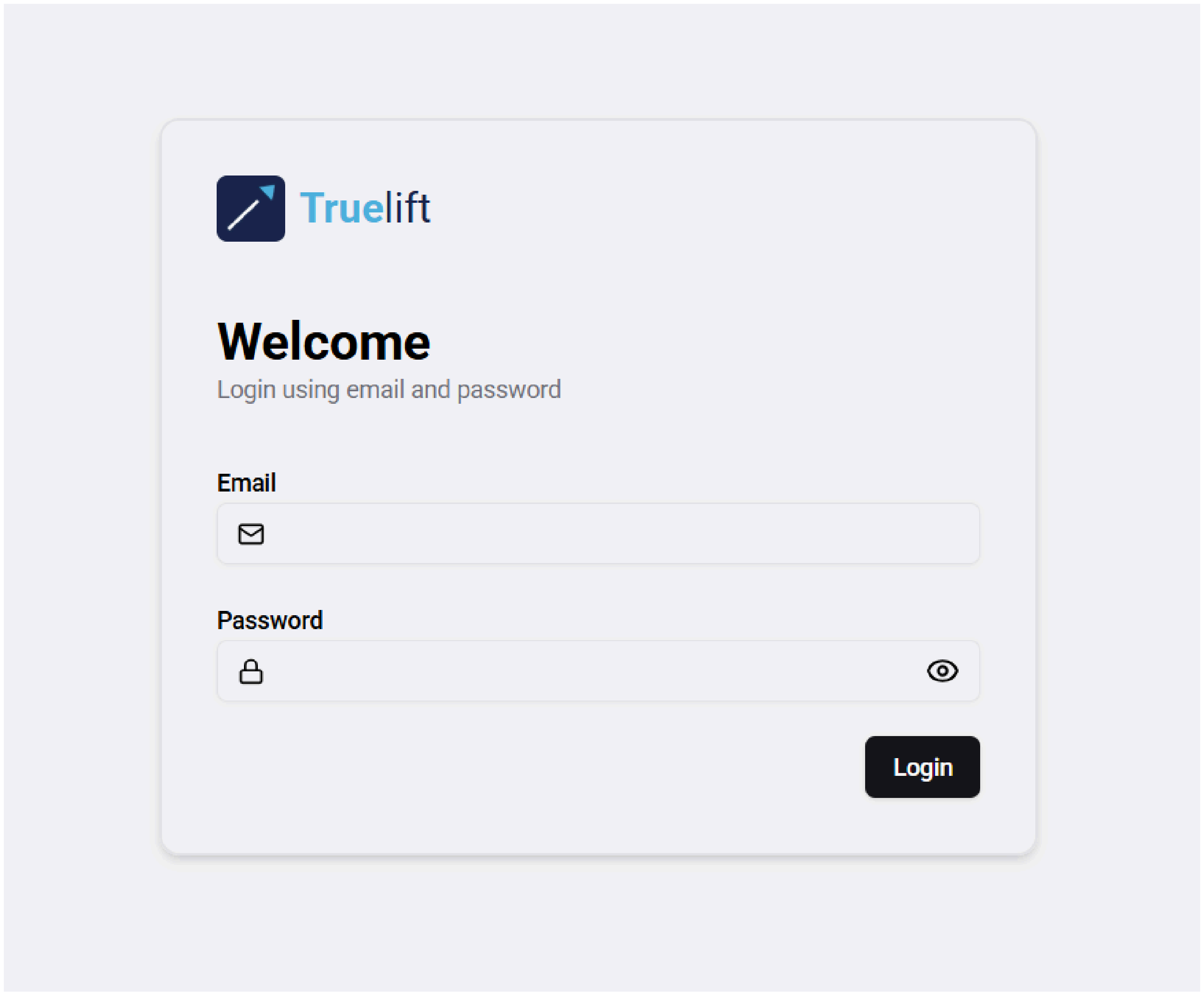
```
const allowed_users_var = new mongoose.Schema({
    email: {
        type: String,
        required: true
    },
    role: {
        type: String,
        required: true
    },
    org: {
        type: String,
        required: true
    },
    tempUserId: {
        type: mongoose.Schema.Types.ObjectId,
        required:true
    }
},{ collection: 'Allowed_users' });



export const Allowed_users_schema = mongoose.model("Allowed_users_schema
```

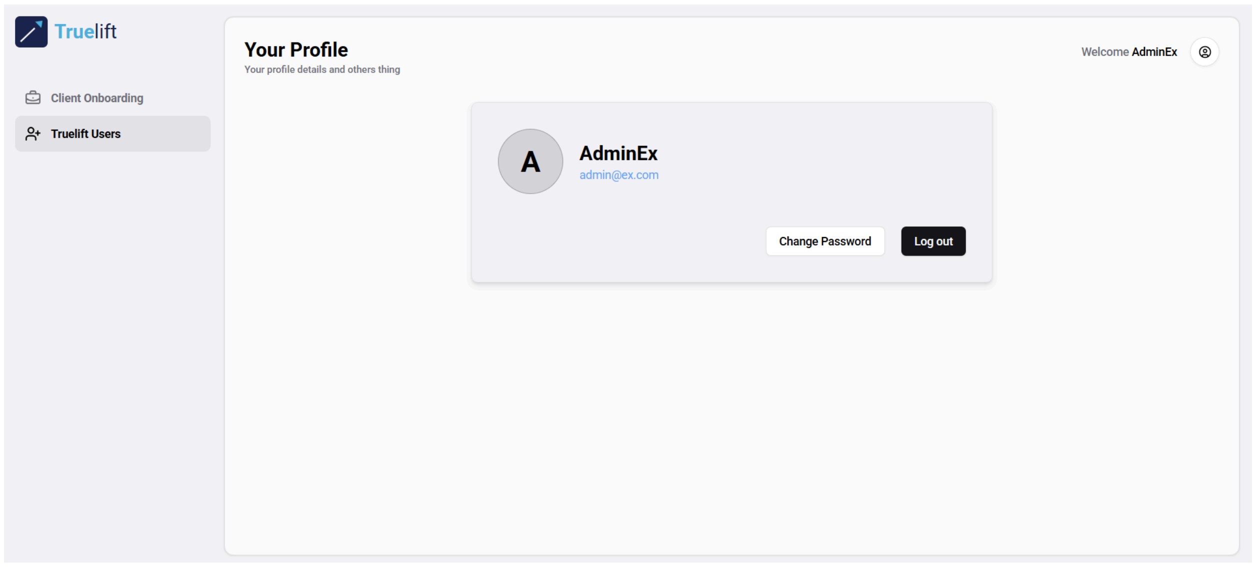
Architecture Overview

Authentication



Welcome

Login using email and password



Truelift Admin

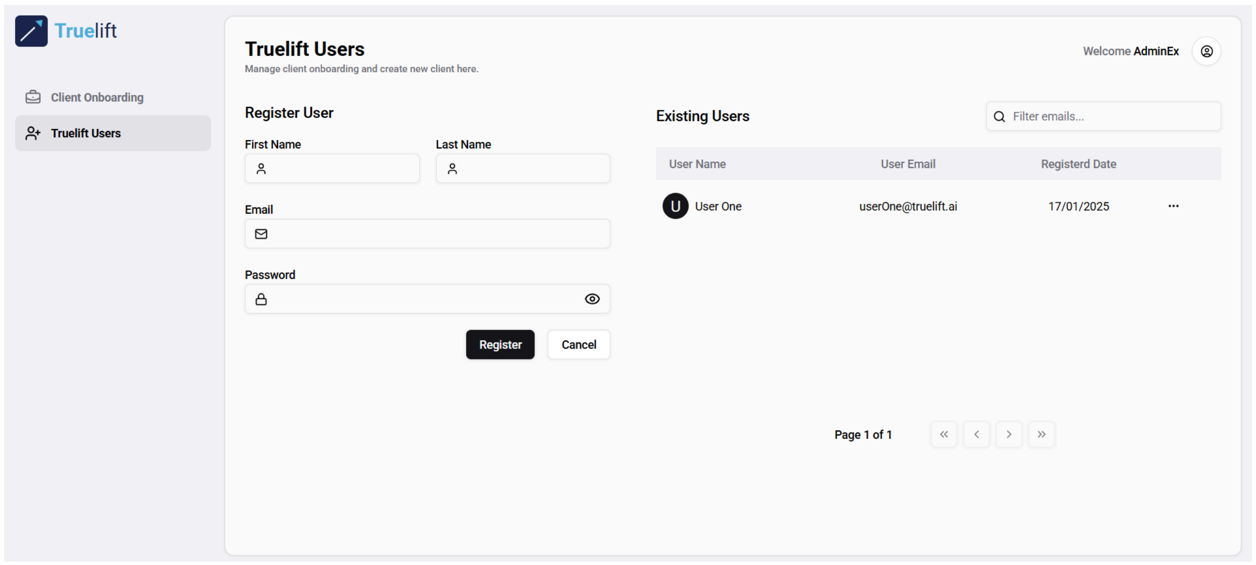
- We will register **Truelift Admin** email on the **Truelift** database → **Truelift_users** schema.
- For the first login **Truelift Admin** have to use the password which is stored in the ENV and set by us.
- If any **Truelift Admin's** email registered in the database he can able to login using ENV stored password.
- This ENV password is safe from others role, because it is only known by who have stored.
- And then **Truelift Admin** can change the password after login from the profile section as shwon in the second image, according to their requirement.
- **Truelift Admin** can now register the **Truelift user** by going to the Truelift user section by setting the there information(explained more in the Truelift user registration section).

Truelift User

- If **Truelift User** already registered **(**in the **Truelift** database → **Truelift_users** schema**)** by **Truelift admin** now user can login using the credentials(email and password) provided by admin.

- **Truelift User** can also change there password after login from the profile section as shwon in the second image.

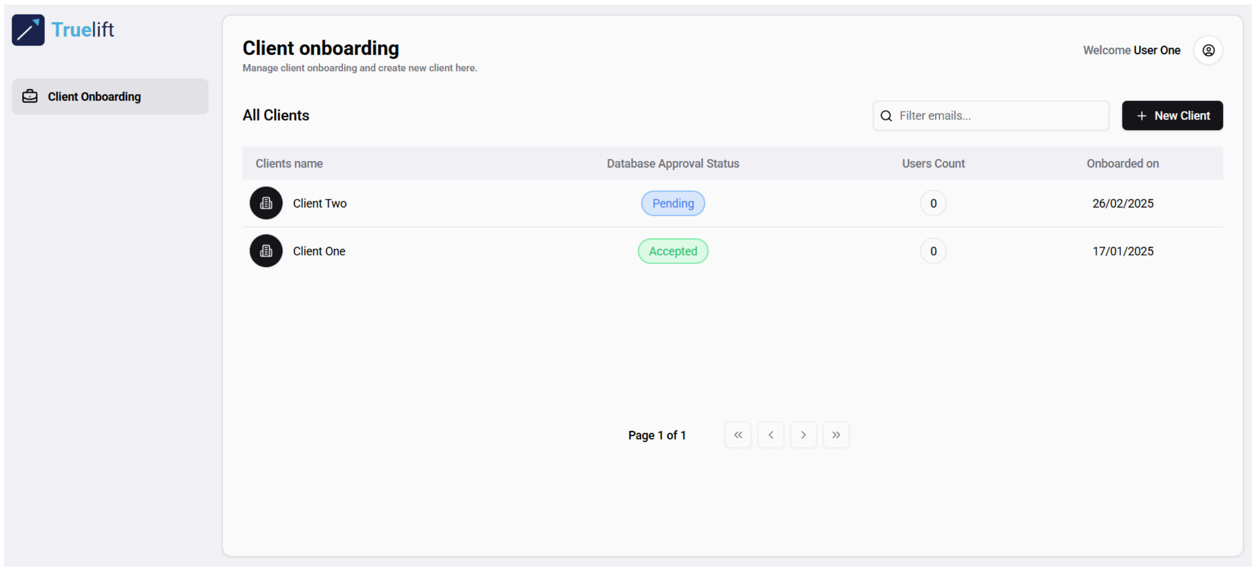
Truelift Users Registration (Truelift admin only can use)



User Name	User Email	Registerd Date
User One	userOne@truelift.ai	17/01/2025

- When **Truelift Admin** Logged in, he get a sidebar tab called **Truelift Users** from where admin can register new **Truelift User.**
- He has to set a initial password to register the user and this email and password further use by **Truelift User** at the time of login
- **Truelift Admin** can also see already registerd user list from the existing users section as shown in the image.

Client creation


Truelift

Client Onboarding

Client onboarding

Manage client onboarding and create new client here.

Welcome User One

All Clients

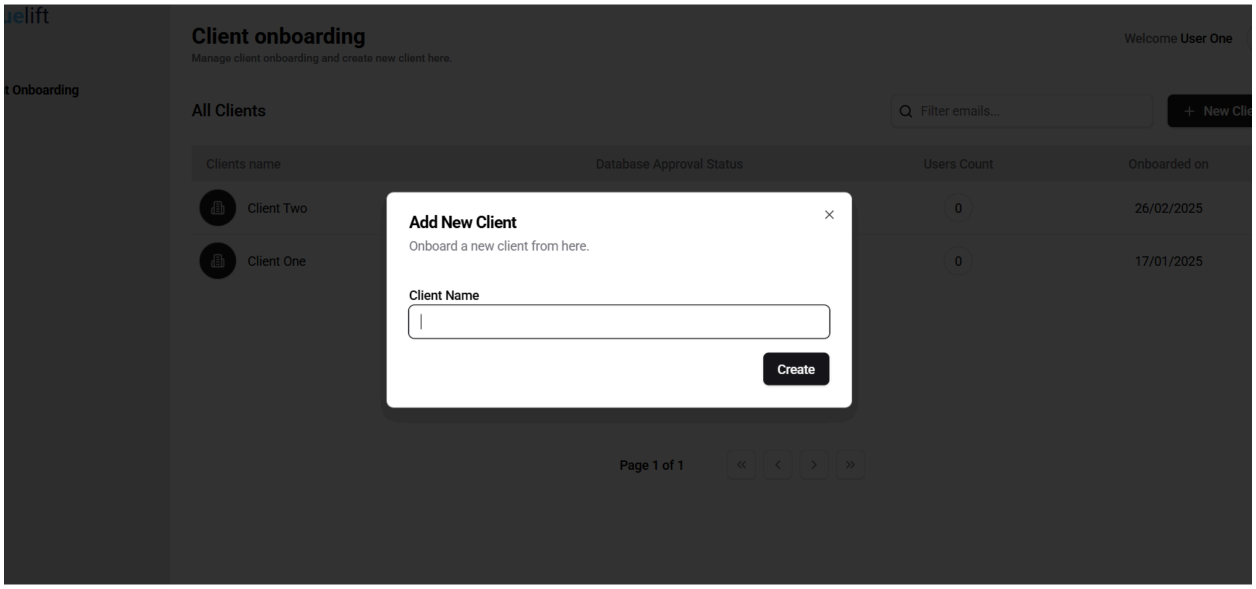
Filter emails...

New Client

Clients name	Database Approval Status	Users Count	Onboarded on
Client Two	Pending	0	26/02/2025
Client One	Accepted	0	17/01/2025

Page 1 of 1

- After logged in **Truelift users and Truelift Admin** can see all already onboarded clients from this page.
- Where they can see details about Client name, Database creation status, Count of users registered inside the client and onboarded date(creation data) for each clie


Add New Client

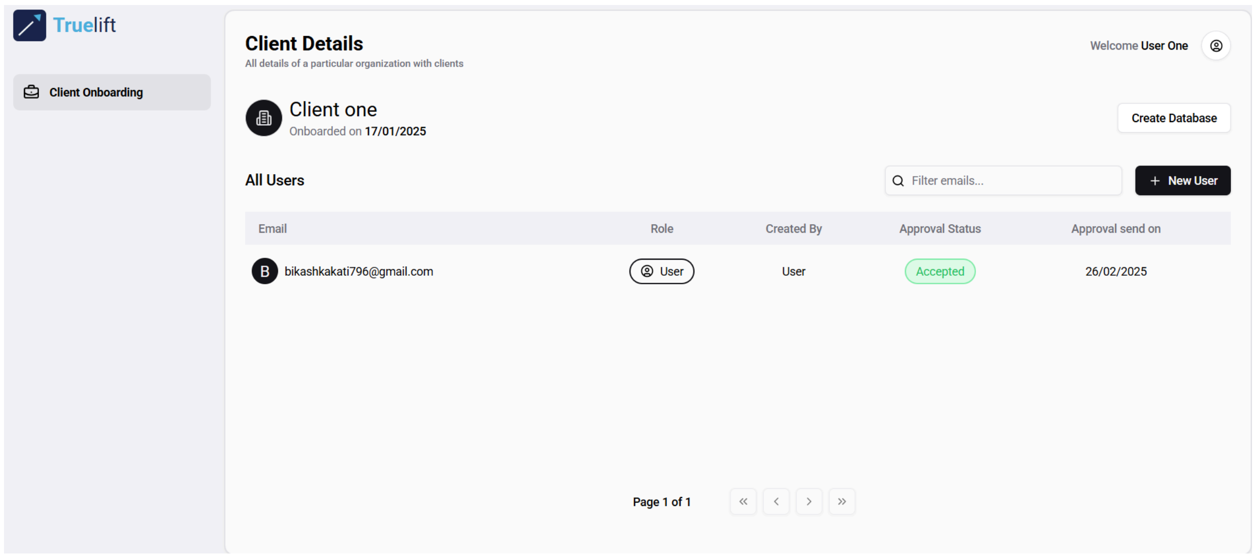
Onboard a new client from here.

Client Name

Create

- The **New Client** feature shown in the image only available to the **Truelift user** that he can create new client.

Database creation and send for approval


Client Details

All details of a particular organization with clients

Welcome User One

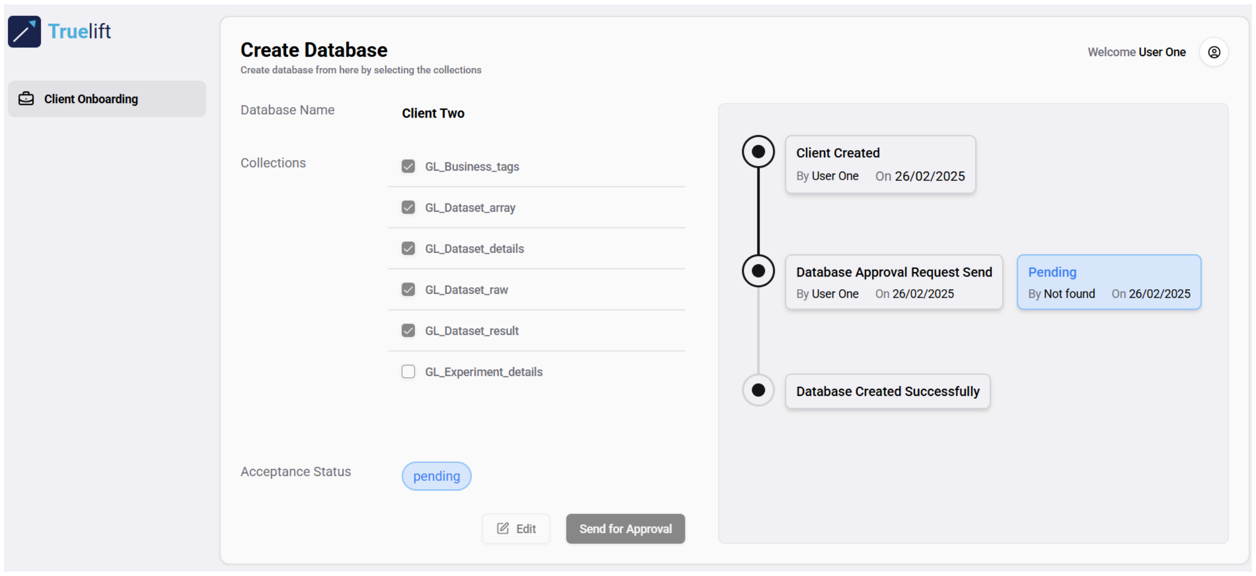
Client one

Onboarded on 17/01/2025

All Users

Email	Role	Created By	Approval Status	Approval send on
bikashkakati796@gmail.com	User	User	Accepted	26/02/2025

Page 1 of 1


Create Database

Create database from here by selecting the collections

Welcome User One

Database Name: Client Two

Collections:
- [✓] GL_Business_tags
- [✓] GL_Dataset_array
- [✓] GL_Dataset_details
- [✓] GL_Dataset_raw
- [✓] GL_Dataset_result
- [ ] GL_Experiment_details

Acceptance Status: pending

Client Created — By User One — On 26/02/2025

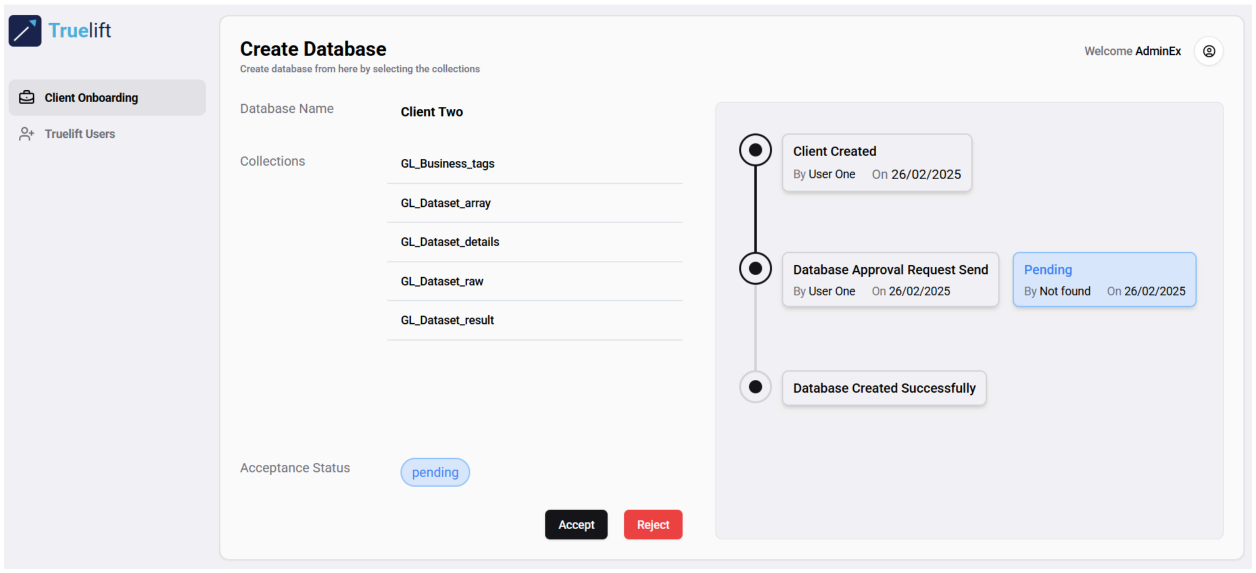
Database Approval Request Send — By User One — On 26/02/2025

Pending — By Not found — On 26/02/2025

Database Created Successfully

- On the client details page which is shown in the first image of the above we have to create database button which only available for the **Truelift User**, for **Truelift Admin** it is show as approve database.
- The second image as shown in above, from there **Truelift User** can create database for client and sent for approval to the admin.
- Database name takes from the client name which we have add at the time of client creation.

- Initially it will show as pending.



Truelift

Client Onboarding
Truelift Users

Create Database
Create database from here by selecting the collections

Welcome AdminEx

Database Name	Client Two
Collections	GL_Business_tags
	GL_Dataset_array
	GL_Dataset_details
	GL_Dataset_raw
	GL_Dataset_result
Acceptance Status	pending

Accept Reject

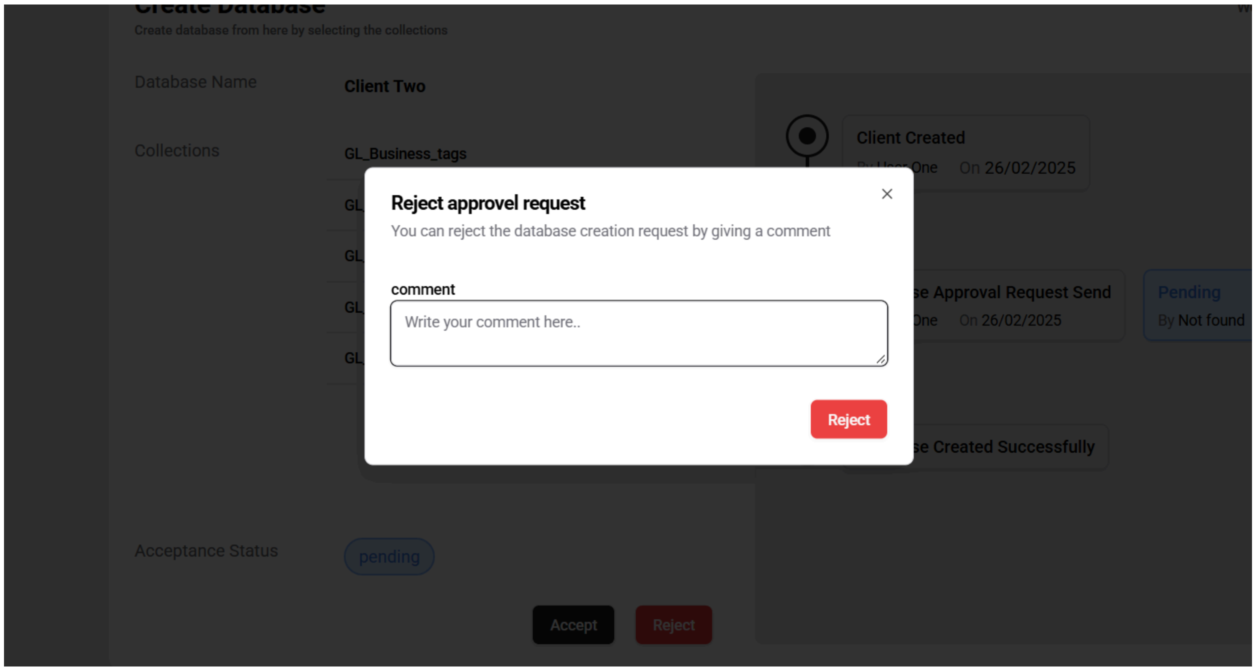
Client Created
By User One On 26/02/2025

Database Approval Request Send
By User One On 26/02/2025

Pending
By Not found On 26/02/2025

Database Created Successfully

- For **Truelift Admin** this ui show to check the database creation details which is sent by **Truelift User** and he can **Accept**/ **Reject** from here.

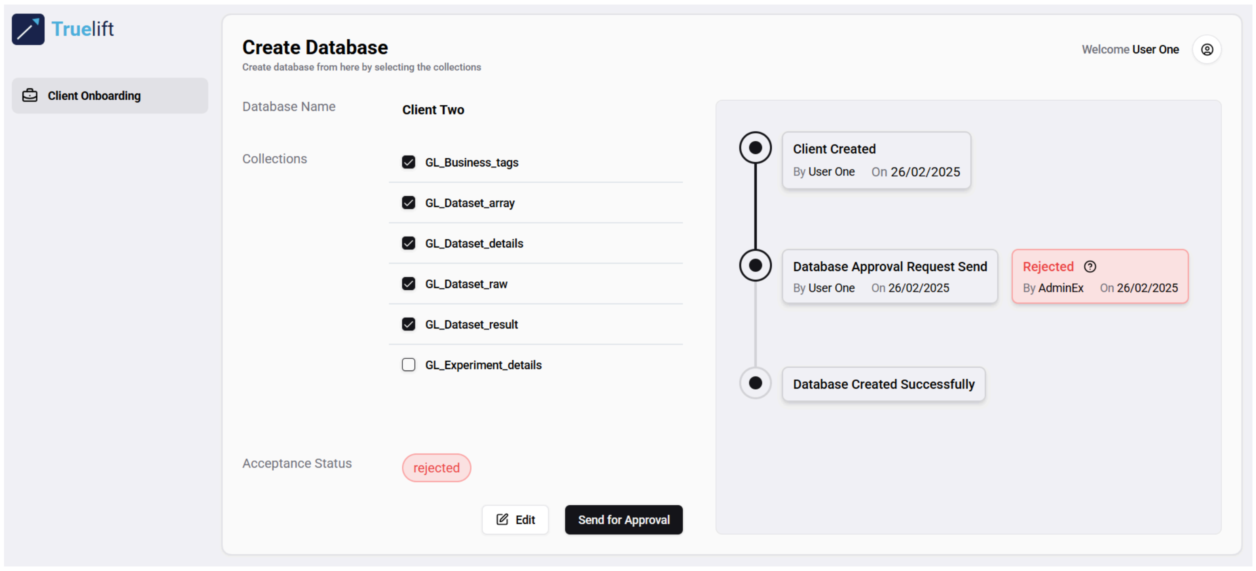


Reject approvel request
You can reject the database creation request by giving a comment

comment
Write your comment here..

Reject

- If **Truelift Admin** want to reject the database creation he need to add a comment and it will reflect on the **Tracking ui**.



Truelift

Client Onboarding

Create Database

Create database from here by selecting the collections

Welcome User One

Database Name: Client Two

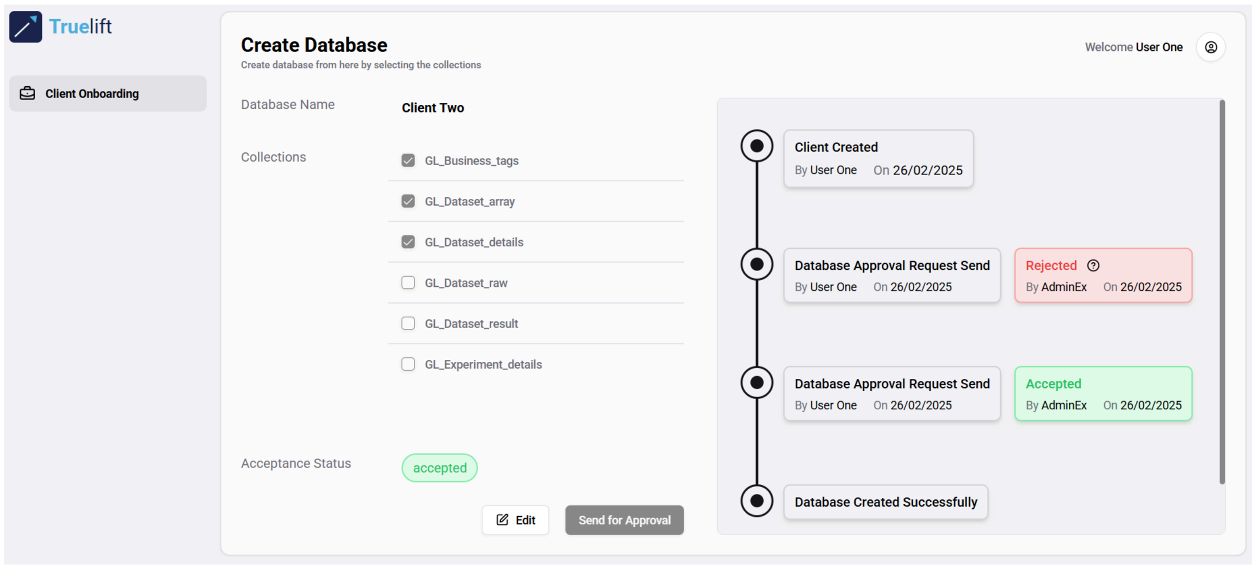
Collections:
- ✓ GL_Business_tags
- ✓ GL_Dataset_array
- ✓ GL_Dataset_details
- ✓ GL_Dataset_raw
- ✓ GL_Dataset_result
- GL_Experiment_details

Acceptance Status: rejected

Edit | Send for Approval

- Client Created — By User One — On 26/02/2025
- Database Approval Request Send — By User One — On 26/02/2025 | Rejected — By AdminEx — On 26/02/2025
- Database Created Successfully

- After Rejection from both side the **Tracking history** ui will show all details.
- Now **Truelift User** again can Update the details and send for another approval Request.



Truelift

Client Onboarding

Create Database

Create database from here by selecting the collections

Welcome User One

Database Name: Client Two

Collections:
- ✓ GL_Business_tags
- ✓ GL_Dataset_array
- ✓ GL_Dataset_details
- GL_Dataset_raw
- GL_Dataset_result
- GL_Experiment_details

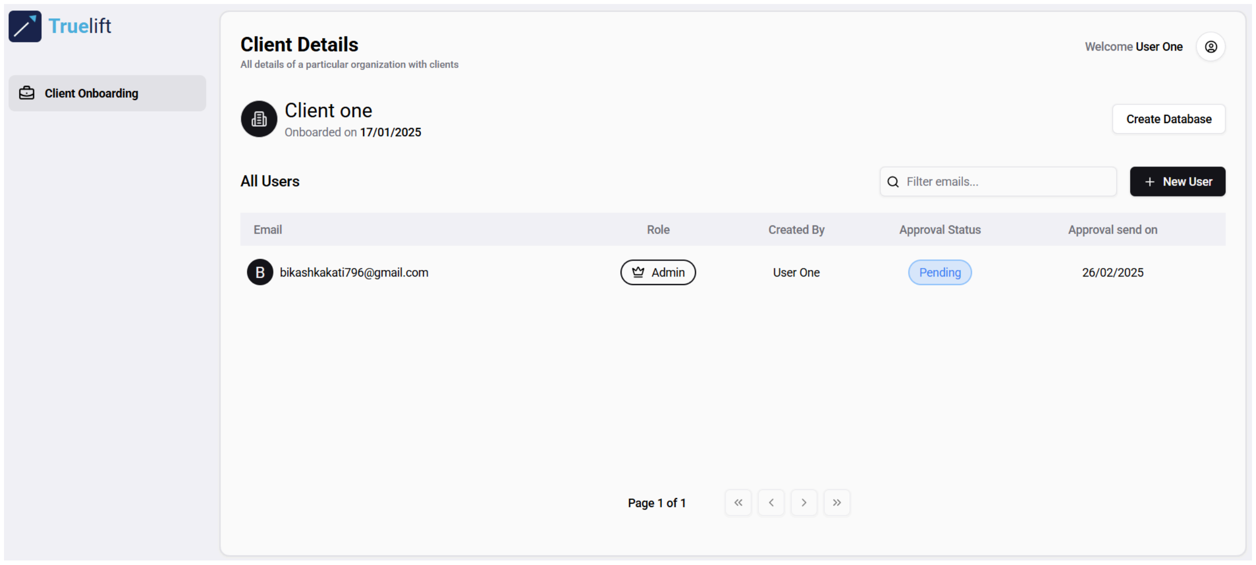
Acceptance Status: accepted

Edit | Send for Approval

- Client Created — By User One — On 26/02/2025
- Database Approval Request Send — By User One — On 26/02/2025 | Rejected — By AdminEx — On 26/02/2025
- Database Approval Request Send — By User One — On 26/02/2025 | Accepted — By AdminEx — On 26/02/2025
- Database Created Successfully

- If Database request accept by **Truelift Admin** the tracking system will show like this, and **Truelift User** has another feature called **Edit,** using this feature user can again send updated database details to the admin for approval.

User creation and sent for approval



Client Details
All details of a particular organization with clients

Welcome User One

Client one
Onboarded on 17/01/2025

Create Database

All Users

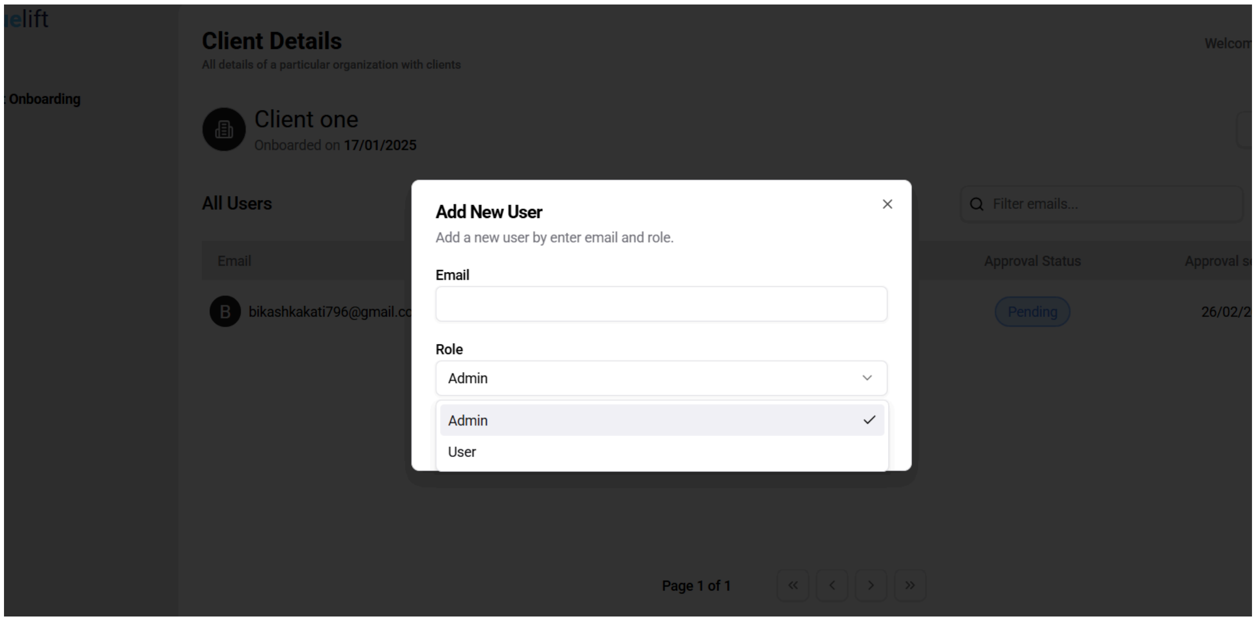
Filter emails...

New User

Email	Role	Created By	Approval Status	Approval send on
bikashkakati796@gmail.com	Admin	User One	Pending	26/02/2025

Page 1 of 1

- **Truelift User** and **Truelift Admin** can see the already existed users from client details page.
- The email, role (Admin, User) and who has created that user and approval details and approved date etc.
- For **Truelift User** a new user creation feature is here as shown in figure, this feature only available for **Truelift User.**



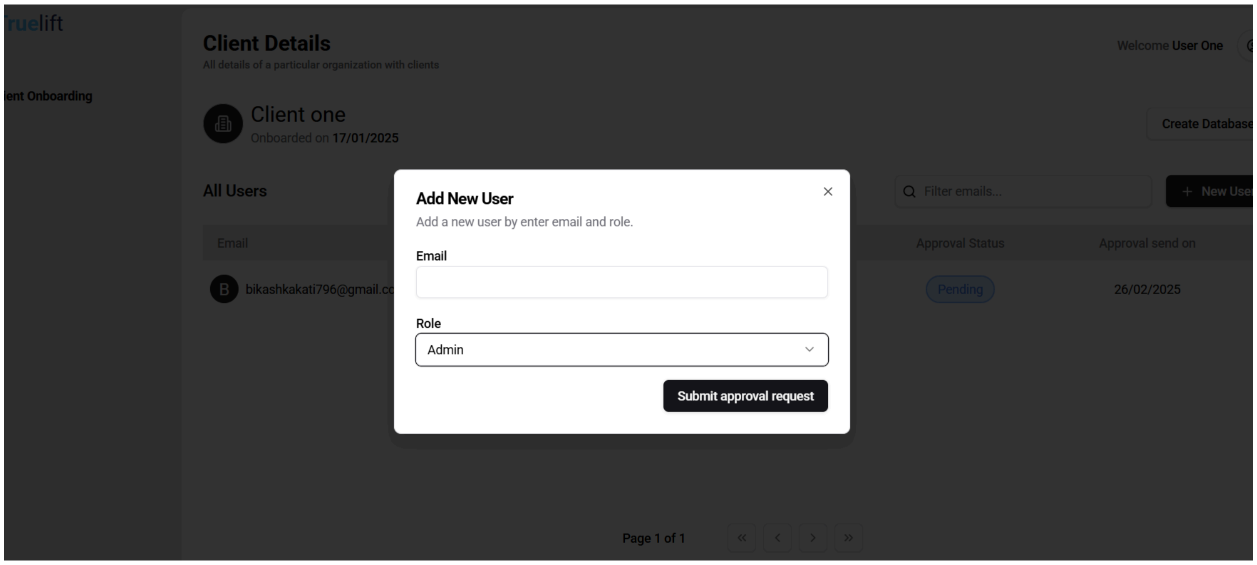
Add New User
Add a new user by enter email and role.

Email

Role
Admin

Admin ✓
User

- **Truelift User** can create Users using this ui by entering user email and selecting the role.



Client Details

All details of a particular organization with clients

Welcome User One

Client one

Onboarded on 17/01/2025

Create Database

All Users

Email	Approval Status	Approval send on
bikashkakati796@gmail.co	Pending	26/02/2025

Page 1 of 1

Add New User

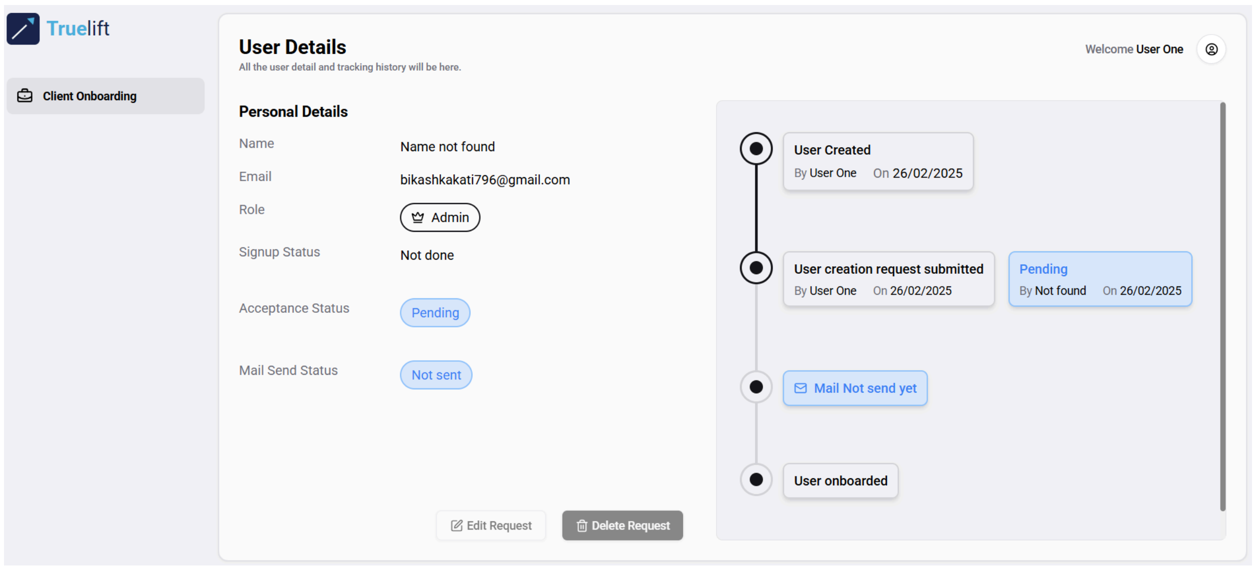
Add a new user by enter email and role.

Email:

Role: Admin

Submit approval request

- By clicking on the submit approval button he can send it for the approval to the **Truelift admin**



User Details

All the user detail and tracking history will be here.

Welcome User One

Personal Details

Name	Name not found
Email	bikashkakati796@gmail.com
Role	Admin
Signup Status	Not done
Acceptance Status	Pending
Mail Send Status	Not sent

User Created — By User One On 26/02/2025

User creation request submitted — By User One On 26/02/2025

Pending — By Not found On 26/02/2025

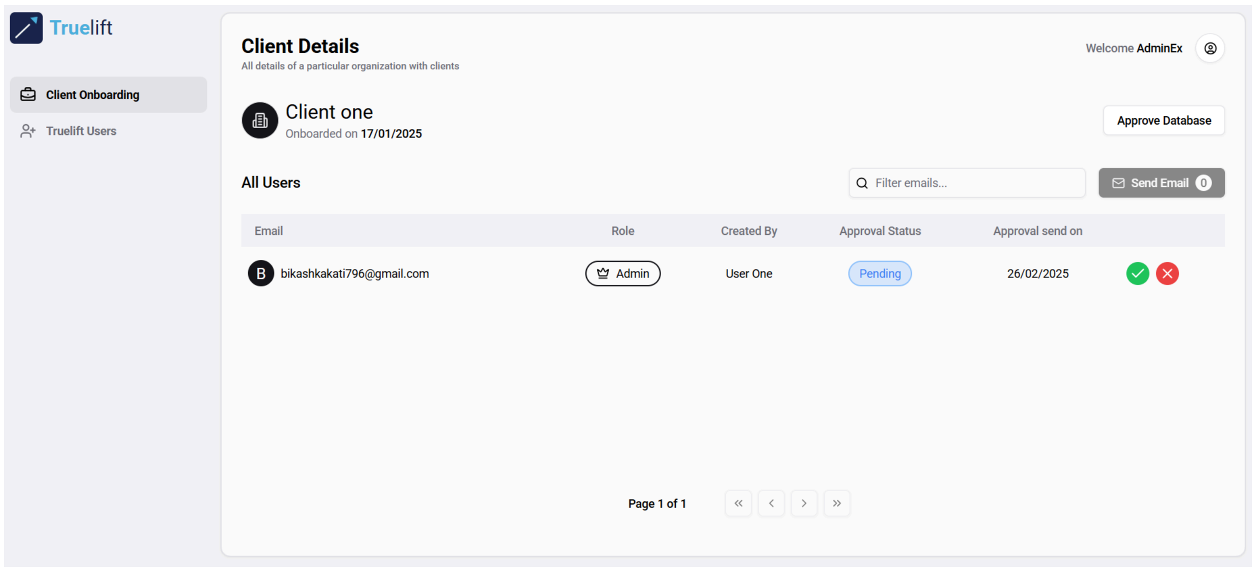
Mail Not send yet

User onboarded

Edit Request

Delete Request

- This is the user details page from where we can see the all details of the user and approval details and everything. (This is the view from **Truelift User**)


Truelift

Client Onboarding

Truelift Users

Client Details

All details of a particular organization with clients

Welcome AdminEx

Client one

Onboarded on 17/01/2025

Approve Database

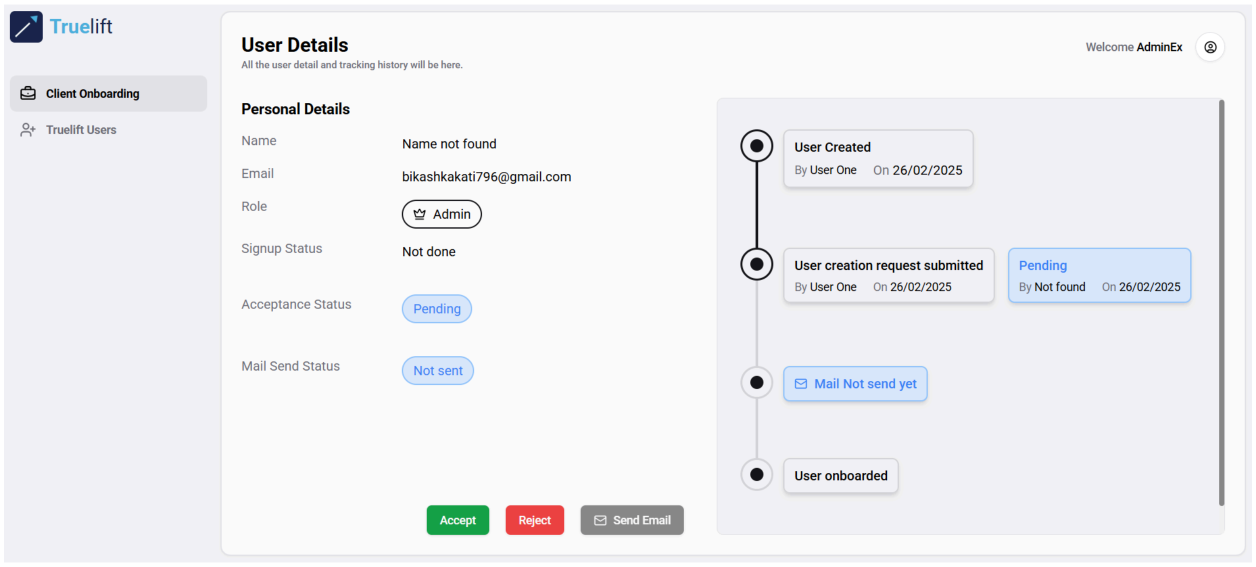
All Users

Filter emails...

Send Email 0

Email	Role	Created By	Approval Status	Approval send on	
B bikashkakati796@gmail.com	Admin	User One	Pending	26/02/2025	✓ ✗

Page 1 of 1


Truelift

Client Onboarding

Truelift Users

User Details

All the user detail and tracking history will be here.

Welcome AdminEx

Personal Details

Name	Name not found
Email	bikashkakati796@gmail.com
Role	Admin
Signup Status	Not done
Acceptance Status	Pending
Mail Send Status	Not sent

User Created
By User One On 26/02/2025

User creation request submitted
By User One On 26/02/2025

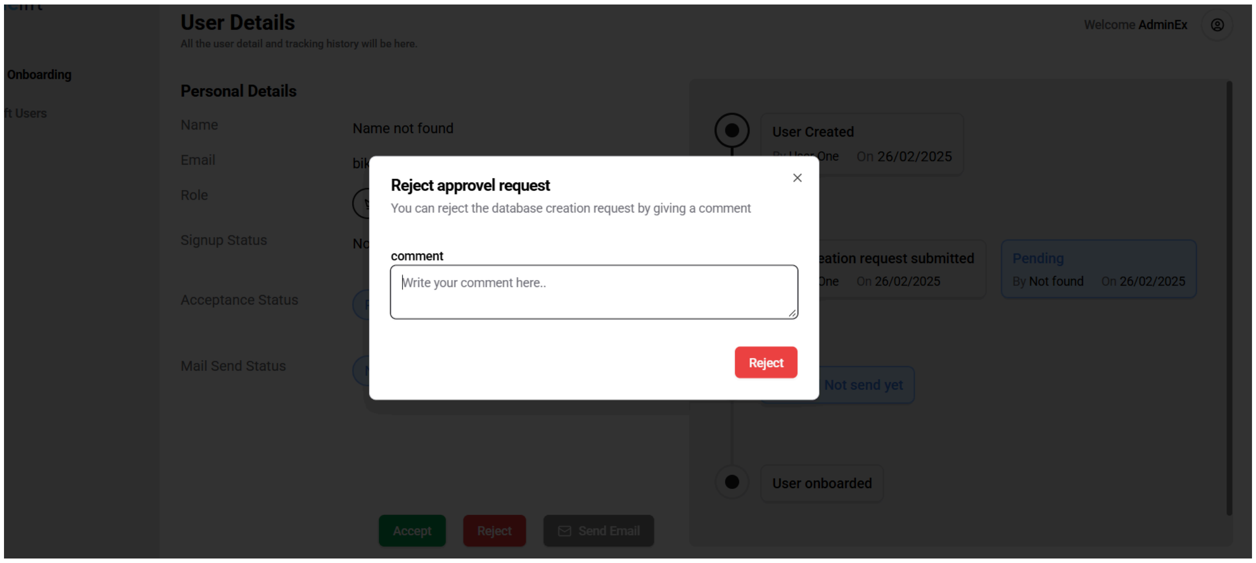
Pending
By Not found On 26/02/2025

Mail Not send yet

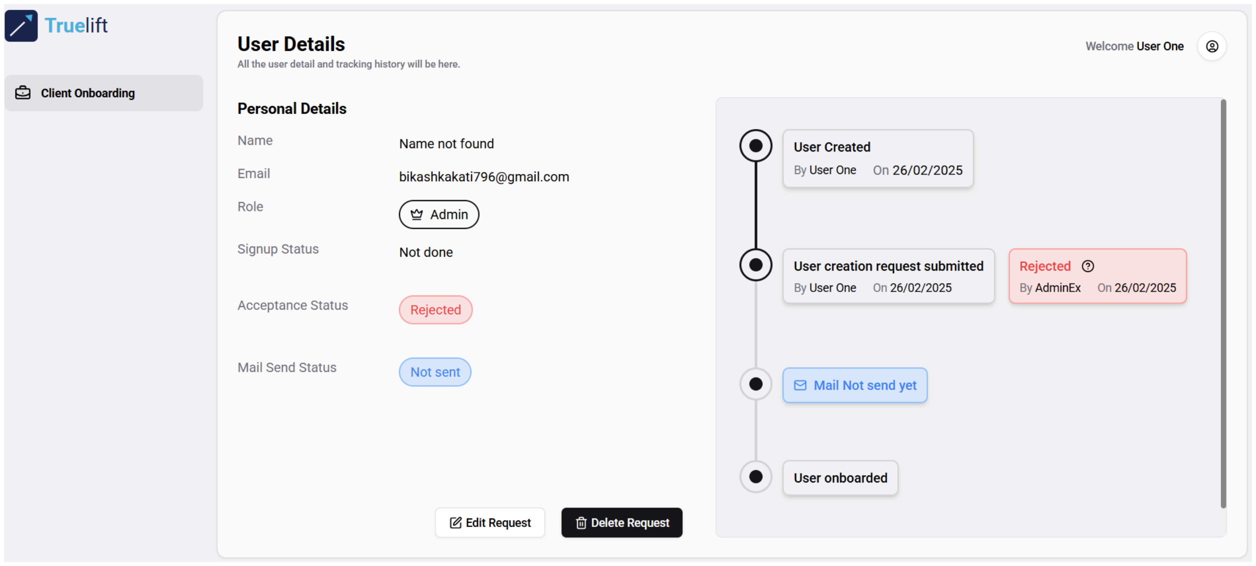
User onboarded

Accept Reject Send Email

- From both page **Truelift Admin** can accept/reject the approval request for user creation.


User Details
All the user detail and tracking history will be here.
Welcome AdminEx
Personal Details
Name: Name not found
Reject approvel request
You can reject the database creation request by giving a comment
comment
Write your comment here..

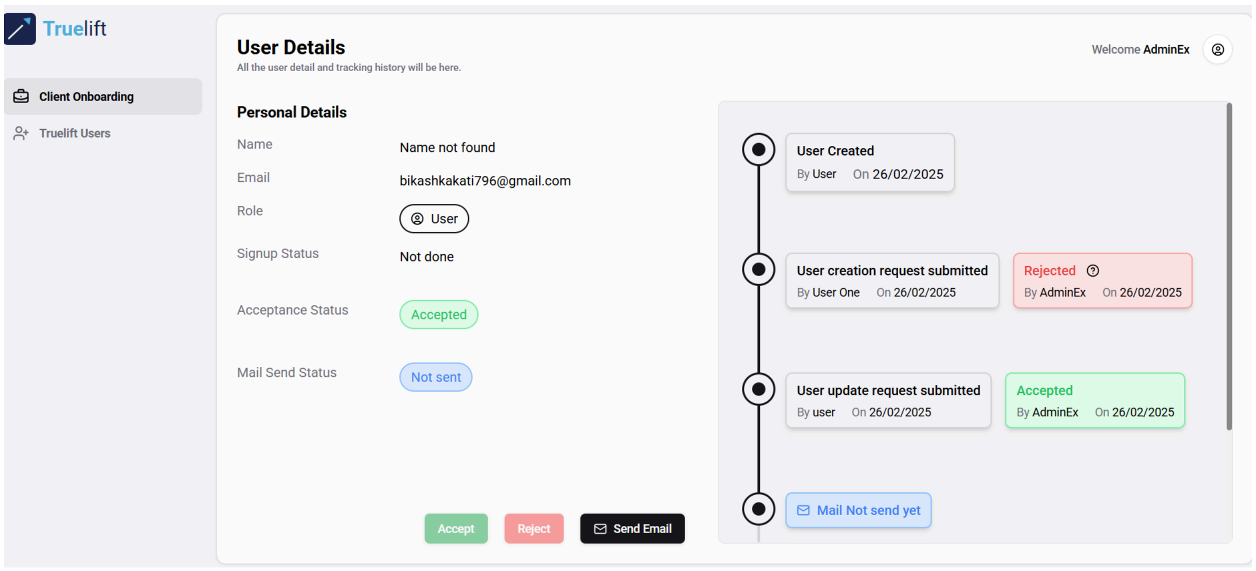
- If he want to reject the request he have to add a comment using this ui and then submit as reference for the **Truelift User**


User Details
All the user detail and tracking history will be here.
Welcome User One
Personal Details

Name	Name not found
Email	bikashkakati796@gmail.com
Role	Admin
Signup Status	Not done
Acceptance Status	Rejected
Mail Send Status	Not sent

User Created — By User One — On 26/02/2025
User creation request submitted — By User One — On 26/02/2025
Rejected — By AdminEx — On 26/02/2025

- If Approval Rejected by **Truelift Admin** then the ui will show like this for the **Truelift User**
- **Truelift User** get the Edit Request again to change for what reason the request rejected by admin.
- After updating those information he can again submit the User creation request.



User Details

All the user detail and tracking history will be here.

Personal Details

Name	Name not found
Email	bikashkakati796@gmail.com
Role	User
Signup Status	Not done
Acceptance Status	Accepted
Mail Send Status	Not sent

User Created
By User On 26/02/2025

User creation request submitted
By User One On 26/02/2025

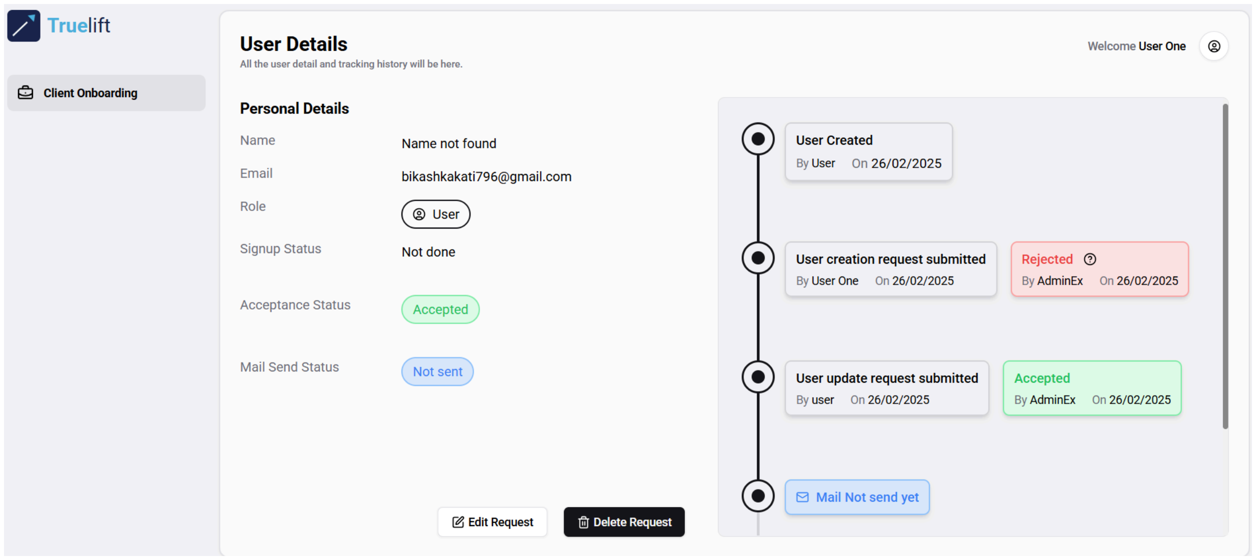
Rejected
By AdminEx On 26/02/2025

User update request submitted
By user On 26/02/2025

Accepted
By AdminEx On 26/02/2025

Mail Not send yet

- If this time **Truelift Admin** accepted the request it will show in the tracking system and and the details.
- Now **Truelift Admin** can send the registration completion mail to the registered user by clicking on the send email button.



User Details

All the user detail and tracking history will be here.

Personal Details

Name	Name not found
Email	bikashkakati796@gmail.com
Role	User
Signup Status	Not done
Acceptance Status	Accepted
Mail Send Status	Not sent

User Created
By User On 26/02/2025

User creation request submitted
By User One On 26/02/2025

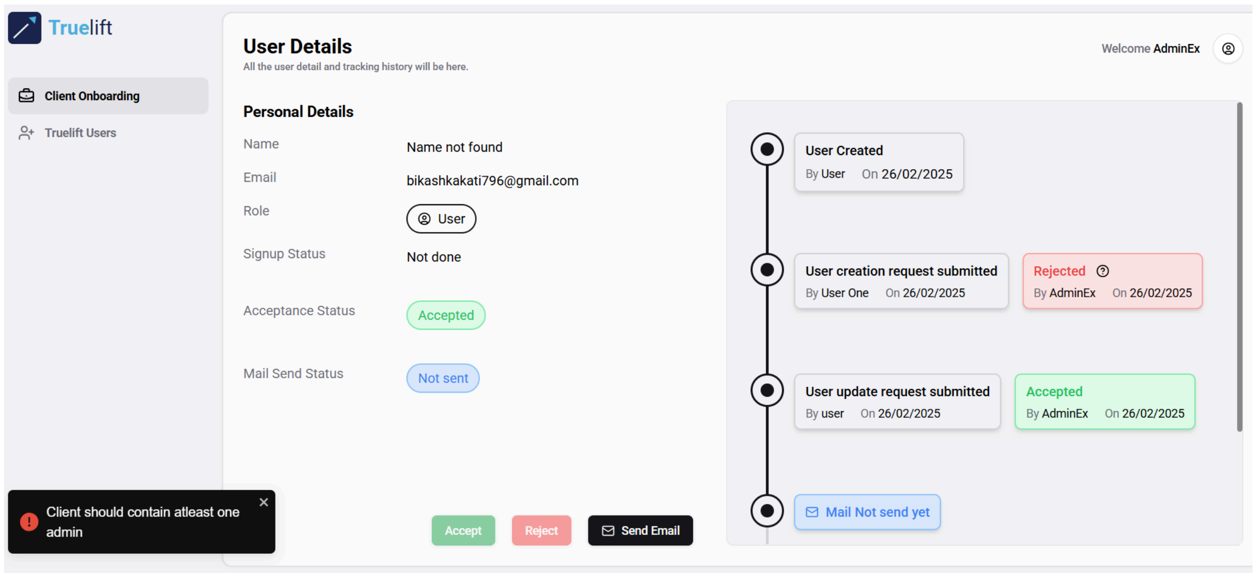
Rejected
By AdminEx On 26/02/2025

User update request submitted
By user On 26/02/2025

Accepted
By AdminEx On 26/02/2025

Mail Not send yet

- **Truelift User** can submit edit and delete request again even after registration complete.



Truelift

Client Onboarding

Truelift Users

User Details

All the user detail and tracking history will be here.

Welcome AdminEx

Personal Details

Name	Name not found
Email	bikashkakati796@gmail.com
Role	User
Signup Status	Not done
Acceptance Status	Accepted
Mail Send Status	Not sent

User Created
By User On 26/02/2025

User creation request submitted
By User One On 26/02/2025

Rejected
By AdminEx On 26/02/2025

User update request submitted
By user On 26/02/2025

Accepted
By AdminEx On 26/02/2025

Mail Not send yet

Client should contain atleast one admin

Accept Reject Send Email

- Send Mail only possible if client has a user with admin role. Otherwise the send mail not work.

Deployment

> **Note:**
>
> The backend is hosted on **AWS Lambda** with **API Gateway**, and the frontend is deployed on **AWS Amplify**.
>
> This backend **uses cookies** for authentication and authorization, requiring specific CORS configurations.

API Gateway Configuration for Cookies

To enable **cross-site cookies**, the following CORS settings must be **explicitly defined** in API Gateway:

```
Access-Control-Allow-Origin: <frontend-url>  # Allow only specific frontend URLs
```

```
Access-Control-Allow-Headers:
  - content-type
  - authorization
  - x-requested-with
  - accept
  - origin
Access-Control-Allow-Credentials: "true"  # Allow credentials (cookies)
```

Api gateway edge cases and consideration

- API Gateway Timeout:
  - Default **30 seconds**.
  - For long-running tasks, consider **asynchronous processing (e.g., SQS, WebSockets, SSE).**

Cookie Settings in Backend

When setting cookies in
login responses, use:

```
res.cookie("userDetails", token, {
  httpOnly: true,    // Prevent access from JavaScript
  secure: true,      // Ensure cookies work only over HTTPS
  sameSite: "none",  // Allow cross-site usage
});
```

Important: If sameSite is not set to “none”, cookies will be **blocked in cross-site requests**

Enviroment Variables on Lambda

```
SERVER_PORT =

ALLOWED_CLIENT =

MONGODB_URL =

ADMIN_PASSWORD =

# JWT
TOKEN_KEY =
TOKEN_NAME =


# MAIL
SMTP_PASS =
SMTP_USER =
SMTP_PORT =
SMTP_HOST  =
```

Lambda CI/CD with github

The following GitHub Actions workflow **automates backend deployment** to AWS Lambda **on every push to the dev2 branch**.

Deployment Script (GitHub Actions)

Create the following folder structure on the git

```
📂 .github
├── 📂 workflows
│    ├── 📝 main.yml
```

Inside
`main.yml`, add this **CI/CD workflow:**

```yaml
name: Deploy to AWS Lambda

on:
  push:
    branches:
      - dev2  # Change if using a different branch

jobs:
  deploy_lambda:
    runs-on: ubuntu-latest
    env:
      CI: true

    steps:
      - name: Checkout Repository
        uses: actions/checkout@v2

      - name: Set Up Node.js
        uses: actions/setup-node@v2
        with:
          node-version: "22"  # Ensure compatibility

      - name: Install Dependencies
        run: npm ci

      - name: Bundle Code & Dependencies
        run: |
          zip -r publish.zip . \
          -x "*.git*" "*.github/*"

      - name: Deploy to AWS Lambda
        run: |
          aws lambda update-function-code \
            --function-name ${{ secrets.AWS_LAMBDA_FUNCTION_NAME }} \
            --zip-file fileb://publish.zip \
            --region ${{ secrets.AWS_REGION }}
        env:
          AWS_ACCESS_KEY_ID: ${{ secrets.AWS_ACCESS_KEY_ID }}
          AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
```

.

Setting Up GitHub Secrets:

For security, store AWS credentials as
GitHub Secrets:

- Navigate to **GitHub Repository** → **Settings** → **Secrets and Variables** → **Actions**
- Click **"New Repository Secret"**, then add:

Secret Name	Description
AWS_LAMBDA_FUNCTION_NAME	The Lambda function name
AWS_REGION	The AWS Region
AWS_ACCESS_KEY_ID	IAM access key
AWS_SECRET_ACCESS_KEY	IAM secret access key

Aws Amplify Env setup

Ensure the **frontend environment variables** are correctly set up in **AWS Amplify**:

```
VITE_BASE_URL = // backend hosted url
```

> **Note:** This product is continuously evolving, and features or configurations mentioned in this documentation may change over time. It is recommended to periodically review and update this document to reflect the latest developments.