Customer Success Platform

Introduction

In today's dynamic business landscape, effective communication and transparency are paramount for ensuring the success of any endeavor. The Customer Success Platform project emerges as a response to this necessity, aiming to revolutionize the way stakeholders are informed and engaged throughout the lifecycle of a project.

Project Overview

The core objective of this project is to develop a robust Customer Success Platform that automates the process of disseminating updates and changes within our system to relevant stakeholders. By harnessing the power of technology, we endeavor to streamline communication channels, ensuring prompt and efficient notification delivery. Ultimately, the platform seeks to enhance collaboration, transparency, and stakeholder satisfaction by keeping everyone abreast of pertinent developments in real-time.

Key Components of Customer Success Platform:

- **Project Description:** An overview of the project's purpose, goals, and objectives sets the stage for understanding its significance and impact.

- **Scope:** Defining the project's boundaries ensures a clear understanding of what falls within the purview of our efforts.

- **Project Stack (Tech):** This section outlines the technological framework of our project, highlighting the utilization of React.js, MongoDB, and Node.js to accomplish our objectives.

- **Escalation Matrix:** A structured hierarchy and process for escalating issues within the project team ensure timely resolution of concerns.

- **Stakeholders:** Identification of key individuals or groups with a vested interest in the project lays the foundation for effective collaboration and communication.

- **Risk Profiling:** Assessing potential risks and devising strategies for mitigation safeguards our project against unforeseen challenges.

- **Phases/Milestones:** Breaking down the project into manageable phases with specific timelines ensures systematic progress and milestone achievement.

- **Sprint-wise Detail:** Detailed insights into each sprint, including timelines, status, and comments, facilitate efficient project tracking and management.

- **Approved Team:** Listing project team members, their roles, and availability ensures clarity regarding responsibilities and resource allocation.

- **Resources:** Identification of requisite resources, including human capital, equipment, and materials, ensures smooth project execution.

- **Client Feedback:** Documenting client feedback, both positive and negative, facilitates continuous improvement and client satisfaction.

- **Minutes of Meetings:** Recording key discussion points from client meetings ensures alignment and accountability across all stakeholders.

Technology Utilized in Project

- ### React.js:

  React.js stands at the forefront of our front-end development stack. Renowned for its component-based architecture and virtual DOM rendering, React.js empowers us to build dynamic and interactive user interfaces with unparalleled ease and efficiency. Its declarative syntax and efficient state management mechanisms streamline development workflows, fostering code reusability and maintainability. By leveraging React.js, we aim to deliver a seamless and responsive user experience across a multitude of devices and browsers.

- ### Node.js:

  Node.js forms the backbone of our back-end infrastructure, enabling server-side JavaScript execution with exceptional performance and scalability. Leveraging event-driven, non-blocking I/O architecture, Node.js ensures optimal resource utilization and responsiveness, making it an ideal choice for real-time applications like ours. Its extensive ecosystem of npm packages grants us access to a myriad of libraries and tools, empowering rapid development and deployment. With Node.js, we can seamlessly handle concurrent connections, execute business logic, and interact with databases, thereby facilitating robust server-side functionality.

- ### MongoDB:

  MongoDB serves as our database of choice, offering a flexible and scalable solution for storing and managing data. As a NoSQL document-oriented database, MongoDB eschews the constraints of traditional relational databases, allowing us to seamlessly handle unstructured and semi-structured data. Its schema-less design facilitates agile development, enabling us to adapt to evolving business requirements with ease. Leveraging features like sharding and replication, MongoDB ensures high availability, fault tolerance, and horizontal scalability, making it an ideal fit for our distributed architecture. By harnessing MongoDB's powerful query language and indexing capabilities, we can efficiently retrieve and manipulate data, thereby empowering data-driven decision-making and analytics.

Features

- ### Login and Role Management using Auth0

  The Login and Role Management feature, powered by Auth0, serves as the gateway to our Customer Success Platform, providing users with secure access and role-based permissions tailored to their organizational roles. This feature streamlines user authentication and authorization processes, ensuring a seamless and secure user experience. Below, we delve into the details of this feature and its functionalities:

  - **Efficient Registration and Authentication:**
    Administrators or auditors can add users directly into the platform, providing their necessary details such as email and name. Upon addition, users automatically receive a welcome email containing their login credentials, including their email and a temporary password.

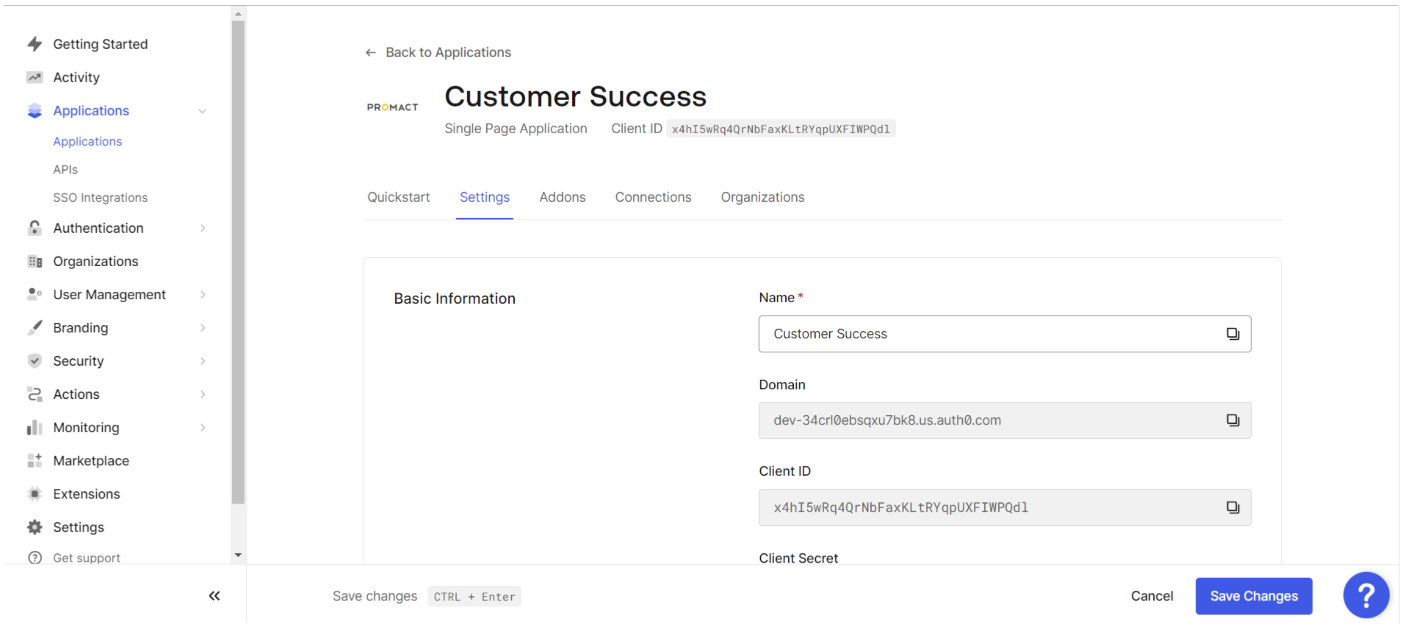
  - **Role-Based Access Control:**
    Auth0's role management functionalities empower administrators to assign specific roles to users based on their organizational positions. These roles determine the level of access and permissions each user has within the platform, ensuring data security and compliance.

  - **Integration with Auth0 Management API:**
    User addition and role assignment are seamlessly automated through Auth0's Management API. This integration streamlines the user onboarding process, ensuring that user profiles and permissions are accurately managed and updated in real-time.

  - **Password Management and Security:**

    Auth0 securely manages user passwords, employing industry-standard encryption techniques to safeguard sensitive information. Users receive their temporary password via email, and they can subsequently reset their password securely if needed, ensuring continuous access to the platform.

  - **Implementation**
    - Created a Single Page Application on Auth0



Customer Success

Single Page Application Client ID `x4hI5wRq4QrNbFaxKLtRYqpUXFIWPQdl`

Basic Information

Name *: Customer Success

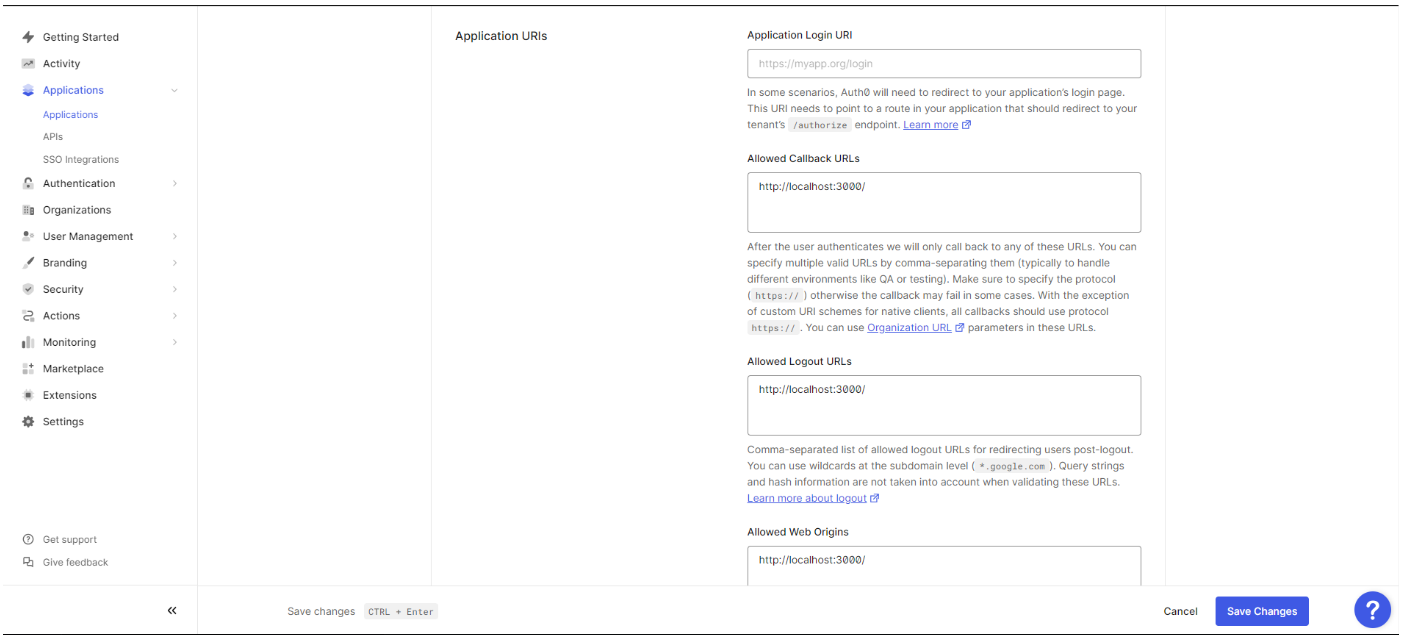
Domain: dev-34crl0ebsqxu7bk8.us.auth0.com

Client ID: x4hI5wRq4QrNbFaxKLtRYqpUXFIWPQdl

Client Secret

Save changes `CTRL + Enter`

- Configured required URLs for Application



Application URIs

Application Login URI

https://myapp.org/login

In some scenarios, Auth0 will need to redirect to your application's login page. This URI needs to point to a route in your application that should redirect to your tenant's `/authorize` endpoint. Learn more

Allowed Callback URLs

http://localhost:3000/

After the user authenticates we will only call back to any of these URLs. You can specify multiple valid URLs by comma-separating them (typically to handle different environments like QA or testing). Make sure to specify the protocol (`https://`) otherwise the callback may fail in some cases. With the exception of custom URI schemes for native clients, all callbacks should use protocol `https://` . You can use Organization URL parameters in these URLs.

Allowed Logout URLs

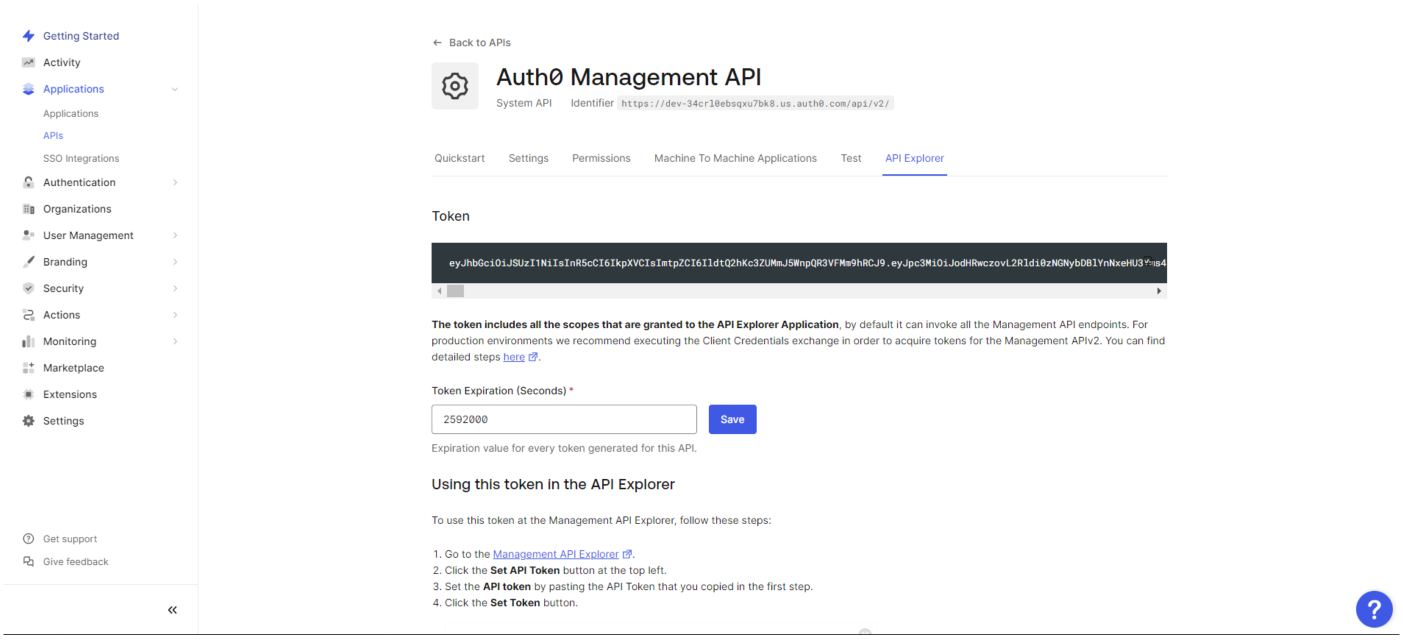
http://localhost:3000/

Comma-separated list of allowed logout URLs for redirecting users post-logout. You can use wildcards at the subdomain level (`*.google.com`). Query strings and hash information are not taken into account when validating these URLs. Learn more about logout

Allowed Web Origins

http://localhost:3000/

- Created Token for Management API



Auth0 Management API

System API Identifier `https://dev-34crl0ebsqxu7bk8.us.auth0.com/api/v2/`

Token

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IldtQ2hKc3ZUMmJ5WnpQR3VFMm9hRCJ9.eyJpc3MiOiJodHRwczovL2Rldi0zNGNybDBlYnNxeHU3Yms4
```

The token includes all the scopes that are granted to the API Explorer Application, by default it can invoke all the Management API endpoints. For production environments we recommend executing the Client Credentials exchange in order to acquire tokens for the Management APIv2. You can find detailed steps here.

Token Expiration (Seconds) *

2592000

Expiration value for every token generated for this API.

Using this token in the API Explorer

To use this token at the Management API Explorer, follow these steps:

1. Go to the Management API Explorer.
2. Click the **Set API Token** button at the top left.
3. Set the **API token** by pasting the API Token that you copied in the first step.
4. Click the **Set Token** button.

- **Endpoints:**
  - **"/login"**
    - **HTTP Methods** = "GET", "POST"
    - **Status Codes** = 201, 400, 409

- **Documentation Reference for Management API**
  - **Create User** = https://auth0.com/docs/api/management/v2/users/post-users
  - **Assign Role to User** = https://auth0.com/docs/api/management/v2/users/post-user-roles
  - **Get Users by Role** = https://auth0.com/docs/api/management/v2/roles/get-role-user

- **CRUD for Each Section of Project:**

  The 15 Sections CRUD feature empowers users to efficiently manage and manipulate various aspects of the Customer Success Platform. Each section corresponds to a critical component of project management, ranging from project budget to client feedback. Here's an overview of the functionality and role-based access control for each section:

  - **Project Budget**:
    - **Role Based Access**
      Create/Update/Delete = Project Managers, Admin
      Read = All Users
    - **Endpoints:**
      - **"/project/:id/public-details"**
      - **HTTP Methods** = "GET", "POST"
      - Status Codes = 200, 500

  - **Version History:**
    - **Role Based Access**
      Create/Update/Delete = Admin, Project Manager
      Read = All Users
    - **Endpoints:**
      - **"/project/:id/version-history"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500

- **Project Description:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/version-history"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Scope:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/project-details"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Project Tech Stack:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/project-details"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Escalation Matrix:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/escalation-metrics"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500

- **Stakeholders:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Auditor
    Read = All Users
  - **Endpoints:**
    - **“/project/:id/stakeholders”**
    - **HTTP Methods** = “GET”, “POST”
    - **Status Code** = 200, 500
- **Risk Profiling:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **“/project/:id/risk-profilling”**
    - **HTTP Methods** = “GET”, “POST”
    - **Status Codes** = 200, 500
- **Phases:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **“/project/:id/phases”**
    - **HTTP Methods** = “GET”, “POST”
    - **Status Codes** = 200, 500
- **Sprint Wise Details:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **“/project/:id/sprint-details”**
    - **HTTP Methods** = “GET”, “POST”
    - **Status Codes** = 200, 500

- **Detailed Timeline Reference:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/project-details"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Approved Team:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/approved-teams"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Resources:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/resources"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Client Feedback:**
  - **Role Based Access**
    Create/Update/Delete = Client
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/project-details"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500

- **MoM of Client Meetings:**
  - **Role Based Access**
    Create/Update/Delete = Admin, Project Manager
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/mom"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500
- **Audit History:**
  - **Role Based Access**
    Create/Update/Delete = Auditor
    Read = All Users
  - **Endpoints:**
    - **"/project/:id/audit-history"**
      - **HTTP Methods** = "GET", "POST"
      - **Status Codes** = 200, 500

Role Based Management

In our Customer Success Platform, role-based management ensures that users have appropriate access levels tailored to their responsibilities. Below are the roles defined along with their respective permissions:

- **Admin Role:**
  - **Full Project Access:** Admins have full access to create, update, read, and delete all projects within the Customer Success Platform.
  - **Section Access:** Admins can create, update, read, and delete all sections of each project in the Customer Success Platform.
  - **User Management:** Admins can manage users by creating, reading, updating, and deleting all stakeholders' accounts.
- **Auditor Role:**
  - **Project Access:** Auditors can create new projects or select existing ones.

- **Project Manager Assignment:** Auditors can assign project managers to projects, facilitating project-specific access for managers.

- **Stakeholder Management:** Auditors can add stakeholders to projects, ensuring their inclusion in the Stakeholders table of the Customer Success Platform.

- **Platform Access:** Auditors can view the Customer Success Platform for all projects.

- **Audit History Comments:** Auditors can add comments to the Audit History table, facilitating documentation and communication within the platform.

Project Manager Role:

- **Platform Access:** Project Managers have access to add, edit, and delete existing content within the Customer Success Platform for their assigned project/s.

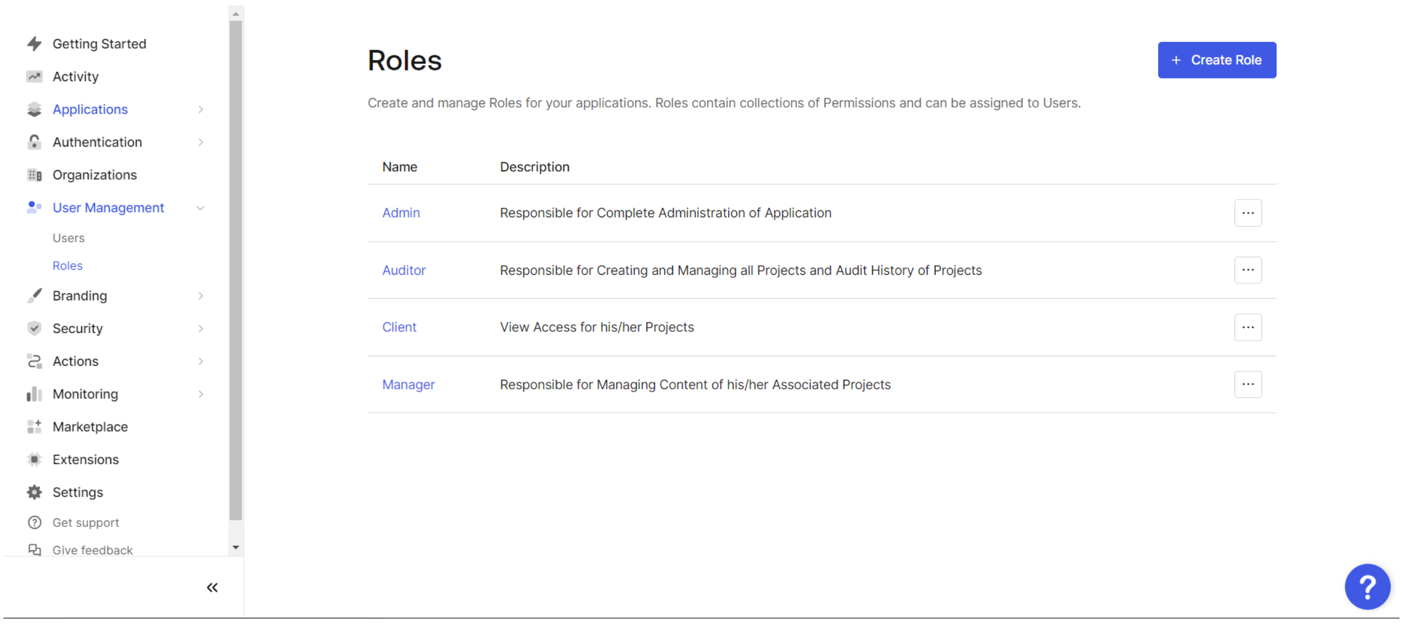
- **Submission:** Project Managers can save and submit updated Customer Success Platform content for their assigned project/s.

Stakeholders/Clients Role:

- **Platform Access:** All stakeholders have view access to the Customer Success Platform for projects assigned to them. This allows them to stay informed and engaged with project updates and information.

Implementation

- Utilized Auth0’s Management API for managing roles of user.
- Created User Roles on Auth0.



Roles

Create and manage Roles for your applications. Roles contain collections of Permissions and can be assigned to Users.

Name	Description
Admin	Responsible for Complete Administration of Application
Auditor	Responsible for Creating and Managing all Projects and Audit History of Projects
Client	View Access for his/her Projects
Manager	Responsible for Managing Content of his/her Associated Projects

- **Email Notification System**

  The Email Notification System is designed to keep stakeholders informed about updates and changes within the Customer Success Platform by sending email notifications in real-time. Below is a detailed description of this feature based on the provided requirements:

  - **Description**
    The Email Notification System serves as a vital communication channel, ensuring stakeholders are promptly notified about any updates or changes within the platform. This feature is responsible for integrating email notification functionality seamlessly into the platform and developing triggers to send notifications for specific events, such as updates to the Audit History table.

  - **Functionality**
    - When updates are made to the Audit History table within the Customer Success Platform, the Email Notification System automatically triggers an email notification to be sent to all stakeholders associated with the project.

    - The email notification contains pertinent information regarding the update, such as the nature of the change, timestamp, and any additional context deemed relevant.

    - Stakeholders receive the email notification in real-time, enabling them to stay informed and engaged with the latest developments within the platform.

  - **Implementation**

  - Utilized **NodeMailer** along with **Google's gmail** service for sending emails.

  - Endpoints:

    - **"/project/:id/sendEmail"**
    - HTTP Methods: "POST"
    - Status Codes: 200, 500

    - /sendEmail/invite
    - HTTP Methods: "POST"
    - Status Codes: 200, 500

- **Export as Document**

  The PDF Document Generation feature enables the creation of comprehensive PDF documents containing all project-related tables and their respective data within the Customer Success Platform. Developed using the **jsPDF** package, this feature provides users with a convenient way to compile and share project information in a structured and easily accessible format. Here's an overview of this feature:

  - **Description:**

    The PDF Document Generation feature empowers users to generate PDF documents dynamically, incorporating all tables and their associated data within the Customer Success Platform. By leveraging the capabilities of the **jsPDF** package, users can seamlessly compile project information into a standardized PDF format, facilitating communication and collaboration among stakeholders.

  - **Functionality:**
    - **Inclusion of All Tables:** The PDF document includes all tables present within the Customer Success Platform, ensuring comprehensive coverage of project-related information.
    - **Dynamic Data Population:** Data from each table is dynamically populated into the PDF document, ensuring accuracy and up-to-date information representation.
    - **Customizable Formatting:** Users have the flexibility to customize the formatting and layout of the PDF document, including styling options such as fonts, colors, and table layouts.
    - **Efficient Compilation:** The generation process is efficient and streamlined, allowing users to quickly compile and download PDF documents with just a few clicks.

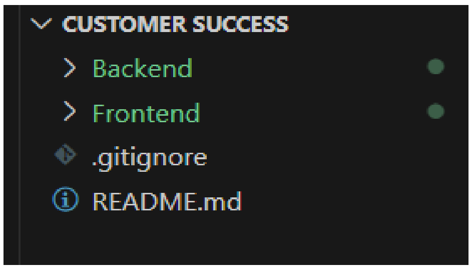
  - **Implementation:**

    The feature is implemented using the **jsPDF** and **jsPDF-autotable** libraries, renowned for their robust functionality and ease of use in generating PDF documents within web applications. Leveraging the capabilities of these libraries ensures compatibility with modern web browsers and provides users with a seamless experience when generating PDF documents within the Customer Success Platform.

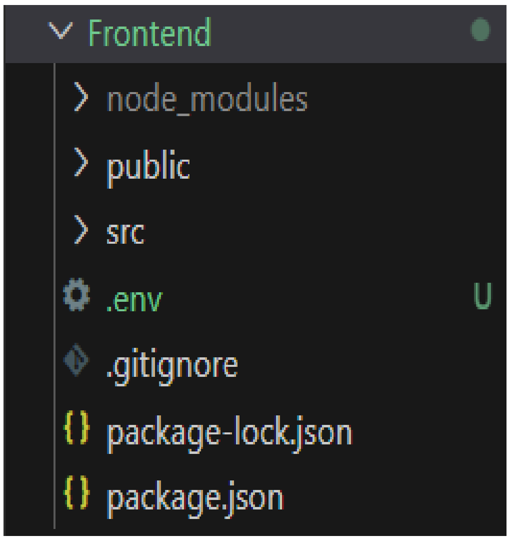
  - **Endpoints:**
    - **"/project/:id/genPDF"**
    - HTTP Methods: "GET"
    - Status Codes: 200, 500

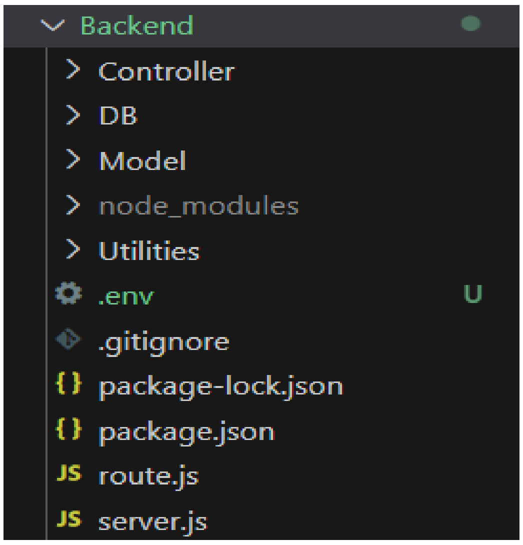
- **Folder Structure:**

Root Folder Structure



Frontend Folder Structure **Backend Folder Structure**





Setup Guide

Follow below steps for setting up the application and using Customer Success Platform:

- **Local Setup**
  - Clone the repository in your local by using below command:

    git clone https://github.com/Jatin-11022002/Promact_Customer_Success.git

- **Backend Setup**
  - **Step 1** = Navigate to Backend folder by using below command in terminal:

    cd Backend

  - **Step 2** = Now run below command for installing dependencies required by application:

    npm install

  - **Step 3** = In Backend folder you will find one file named as "**.env.sample"**, rename the file to **.env**. This step is necessary as all important URL and API key used by the application are stored in this file. You can also use your own values in the file for altering the behavior of application.

  - **Step 4** = run below command to start the backend server:

    node server.js

Now you should be getting **"server started"** and **"db started"** message in console, denoting that backend server and database are connected and running successfully.

- Frontend Setup

  - **Step 1** = Now open another terminal and use below command to navigate to Frontend folder (Assuming to be currently in root folder):

    cd Frontend

  - **Step 2** = Now run below command for installing dependencies required by application:

    npm install

  - **Step 3** = In Frontend folder you will find one file named as **".env.sample"**, rename the file to **".env"**. This step is necessary as all important URL and API key used by the application are stored in this file. You can also use your own values in the file for altering the behavior of application.

  - **Step 4** = Now run below command to start the frontend server:

    npm start

You will be able to see the URL on console, denoting that React server is running successfully. Navigate to that URL in your browser

Now you will be able to use the application.

Credentials for Testing Application

- **Admin:**
  - **Email** = **admin@gmail.com**
  - **Password** = **Jatin@123**

- **Manager:**
  - **Email = manager1@gmail.com**
  - **Password = Jatin@123**

  - **Email = manager2@gmail.com**
  - **Password = Jatin@123**

- **Auditor:**
  - **Email = auditor@gmail.com**
  - **Password = Jatin@123**

  - **Email = manager2@gmail.com**
  - **Password = Jatin@123**

- **Client:**
  - **Email = jatinramchandani15@gmail.com**
  - **Password = Jatin@123**

  - **Email = samarbgmi11@gmail.com**
  - **Password = Jatin@123**