# Base Saas Application

## Overview

This plan outlines the development and implementation of a base SaaS dashboard. The dashboard will serve as a foundation for new portals and services that we plan to launch. The following documentation describes the features, integrations, and technical specifications of the application.

### End User Features

- Authentication: Integrated with AWS Cognito for email and social login authentication.
- Subscription: Users will be able to sign up and pay for services generated by the platform.
- Dashboard: Authenticated users and admins will be able to manage various services and roles within the application.
- Multi-tenancy: Users will be able to create accounts and invite others to their account. Users can have access to multiple accounts, each with different roles.
- Access Control: Users will have roles that control the level of access within an account.

### Terminology

- Account: An account provides a data structure to group companies and the users that consume our services. Each account is billed for the services that are consumed.
- Invoice: Accounts can review and receive invoices for services they consume. A customer must pay an invoice to continue using the platform.
- User: A user is someone that can login to the platform using an email/password.
- Role: A role is an intersection between a account, or many, and the level of access within that account.
- Super Admin: A user that has access via a role to all data within all accounts in the platform.
- Admin: A user that has access via a role to all data within an account.

### Integrations

- Stripe: Payment support and PCI compliance.
- Google: Third party authentication.

## Technical Specifications

### Runtime

- Node.js v19.4.0

### Software Frameworks

- React v18.2.0
- Tailwind v3.0

### Infrastructure

The platform will be hosted on AWS using the following services. Our goal is to create an entirely stateless platform to lower hosting costs and improve performance. AWS services will include:

- Cognito: Supports for authentication and access control.
- IAM: Defines access control.
- CodeCommit: Primary code storage mechanism.
- S3: Hosting for uploaded assets and build from our front end.
- Lambda: Server side functions, resolvers, and mutations.
- DynamoDB: Data storage specific to application business logic.
- AppSync: Provides a managed GraphQL to interface backend functionality.
- SES: Email integration.

### Build Tools

- Amplify: Amplify CLI will manage deployment of core resources such as Cognito, AppSync, DynamoDB, and S3. It will also control the build tools to deploy the frontend.
- Serverless: For the backend, we will leverage the Serverless Framework v3. There are certain shortcomings in Amplify that can be avoided. These shortcoming include a lack of support for complicated cloud formations as well as typescript based lambda build tools.
- Next.js: Allows us to prerender aspects of the application for performance.
- Typescript: Typescript will allow us to write reusable code across various platforms.

## UI/UX

The base platform will be designed to be flexible in terms of style, branding, color, typography, and fonts. This will enable us to easily tailor the platform to the specific needs of each business. The following section describes the major component and layout of the platform.

### Layout

These are the top level components of the platform.

- Top navigation bar (Unauthenticated): Transparent header with logo and informational tabs.
- Top navigation bar (Authenticated): Sectioned header with logo, authenticated profile drop down, notifications icon, and a mechanism to switch between roles assigned to the user.
- Bottom footer bar (Authenticated & Unauthenticated): Informational links and directory for users that is provided on every page within the application.

- Sidebar (Authenticated): Displays enabled modules and services relating to the users current access and scope.

### Authentication

**Pages to support unauthenticated users**

- Sign-up page: Allows users to create a new account by providing their personal information and creating a password.
- Sign-in page: Allows users to log into their account using their email or username and password.
- Forgot password page: Allows users to reset their password if they've forgotten it.
- Multi-Factor Authentication (MFA) page: Allows users to set up and configure MFA for their account.

**Pages to support authenticated users**

- Profile page: Allows users to view and update their personal information and account settings.
- Security settings page: Allows users to manage their account's security settings, such as password complexity requirements and session duration.

**Pages to support administration of users**

- User management page: Allows administrators to manage the users within their Cognito user pool, including creating, updating, and deleting user accounts.
- Group management page: Allows administrators to create and manage groups within the user pool, and assign users to those groups.

### Application Support

**Template pages to support potential uses cases within the application**

- Dashboard Page:  A default page with example graphs and typical UI that a dashboard would require.
- Empty Form Page: An empty page to serve as a template for forms.

### Billing

- Subscriptions Page: List subscriptions.
- Billing History Page: List paid and unpaid invoices.
- Manage Credit Cards Page: List and manage credit cards on file.

## Resources and Reference

- AWS Amplify Serverless Plugin
- Using Serverless Framework with the Amplify Client Library

- Deploy a Next.js 13 app to AWS with Amplify Hosting

## Additional Topics

Data Backup and Recovery: It is important to include a plan for data backup and recovery in case of any data.

## Currently Project Status

- React has been installed