- Infrastructure description:

  The Project is made using:
  - Angular for Front-end
  - Node.js for Back-end
  - PostgreSQL for Database
  - Developed using TypeScript

  The Services used to Host/Create:
  - S3 Buckets for hosting the front-end
  - Elastic Beanstalk for hosting the back-end
  - RDS for creating a database instance

  The Configurations for Services:
  - S3 Buckets is set to Public and for static website hosting
  - EB environment are using Node.js 16 Node.js 16 running on 64bit Amazon Linux 2/5.6.1
  - RDS database is set to public with a TCP connection

- App dependencies:
  1. Front-end:

     Angular: a Front-end framework

     Ionic: An open source mobile UI toolkit for building cross-platform mobile apps from a single code base in Angular.

     Core: Modular standard library for JavaScript

     RxJS: Reactive Extensions For JavaScript

     Zone.js: Implements Zones for JavaScript, inspired by Dart.
  2. Back-end:

     aws-sdk: AWS SDK for JavaScript

     bcryptjs: A library to help you hash passwords.

     body-parser:Parse incoming request bodies in a middleware before your handlers, available under the req.body property.

     cors: a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

     dotenv: Dotenv is a zero-dependency module that loads environment variables from a .env file

     email-validator: A simple module to validate an e-mail address

     express: Fast, unopinionated, minimalist web framework for Node.js.

     jsonwebtoken: adds JWTs to the application

     pg: Non-blocking PostgreSQL client for Node.js.

     sequelize: promise-based Node.js ORM tool for Postgres

     validator:A library of string validators and sanitizers.

- Pipeline process:

  The Pipeline consists of 3 steps:

  1. The building step:
     a) Install Front-End Dependencies
     b) Install API Dependencies
     c) Lint the frontend
     d) Front-End Build from Angular to JavaScript
     e) API Build from TS to JS
  2. The hold step:
     a) This is a transience step that stops the pipeline from being executed to until the Admin approves the deploy step
  3. The deploy step:
     a) Deploys the API to EB
     b) Deploys the Front-end to S3