Technical Documentation

The Tech Stack

Networthy is a personal finance startup focused on helping young alumni achieve financial wellness and freedom. This project involves building a web-based platform that allows customers (alumni) to list their financial goals, view related content (as per goal tags) provided by the coaches, work towards that goal, and progress through the goal stages.

Networthy will be a MVP Web Application, which will support multiple APIs, for functionalities like, Login/Signup, adding a goal, selecting a coach for a specific goal, updating the goal stage(Not started / Inprogress / In review / Resolved), for the client, and for the coach to Login/Signup, to resolve the completion request for a goal raised by the client, etc. Since the Web Application will be built from scratch, we have decided to go with the following tech stack after consulting the client:

FrontEnd:

ReactJS
HTML5
SAAS
Webpack
Material UI
Redux

We decided to go with ReactJS as it provides reusable components and interactivity to the layout of any UI. It also supports dynamic alteration of the selected user interface upon significant data change. ReactJS is a perfect combination of JavaScript and HTML tags and helps the application deal with a vast set of data. ReactJS also provides excellent cross platform support.

Backend:

Java
SpringBoot

We decided to opt for java because as an platform independent, object-oriented programming language it helps us improve the reusability and flexibility of code and syntaxes. Additionally, Java binds data into a single unit module that restricts intervention and access to the outside world. Java combined with spring-boot is a perfect choice for web application development which offers scalability, security and platform mobility.

DataBase:

MongoDB

We decided to go with MongoDB because of the document oriented storage approach and its ease to scale. MongoDB also provides auto sharding and fast in-place updates.

Server:

Backend :- Apache Tomcat
Frontend :- Node.js

Web Applications

1. Web Application:
    a. Frontend -
        i. Based primarily on React (a JS framework);
    b. Backend Server -
        i. Based primarily on Java (Spring Boot);
    c. Database -
        i. MongoDB is being used as a DB.

Mobile Applications

N/A

AI/ML Applications

N/A

Hardware Applications

N/A

Common considerations

How familiar is the team with the technology?

Every team member brings in a production level full stack development experience with Java and SpringBoot framework as a Backend Language/Framework & Javascript and React as a Frontend Language/Framework. Our current member skill division involves 2 members with more than 2 years of experience in Frontend and the other 3 members with around 3 years of experience in Backend development including the CI/CD deployment through AWS. Moreover, the project includes only minimal third-party library usage including the frontend integration with Calendly inorder to schedule meetings between the NetWorthy client and Coaches.

How common/widespread is the technology?

For the Backend we are using Java with Spring Boot framework, Java is a renowned and influential programming language for development of web applications, Java is currently used by over 83,000 websites, some of the renowned websites using Java are IBM developer works, Airbnb, Spotify Web, etc.
As for the Frontend we are using ReactJS, which is a JavaScript framework used for building user interfaces, React was recently voted the 2nd most famous web framework for professional web development, currently there are 10,883,823 live websites using React, some of the most famous websites using React JS framework are Netflix, Instagram, New York Times, etc.

How well does the technology support your problem statement?

Since the project (problem statement) involves the reuse of multiple frontend components. The best way to ensure the scalability of frontend components and generic/non-repeating code is to follow the Web Components based standard.

Keeping these points in mind, choosing React (a Javascript framework) saves us time and keeps the frontend code scalable and involves less clutter as it closely follows the Web Components standards. Moreover, we choose SASS as a preprocessor to ensure we write generic style CSS code using mixins and imports for the frontend design.

Do you even need that technology at all?

We have ensured to keep the third party integrations to the very minimum inorder to not fall into the pit of frequent security flaws, depreciating modules and what not. Only third party integration we expect is of calendly (that too not through importing its module). Apart from that, the only dependencies that will be used within the frontend involves, react, react-dom, webpack, node-sass, react-scripts which all are very well maintained and get shipped together in a Create-React-App (CRA) application. We have made sure to not import/include unnecessary modules into the project and keep the approach as simple as possible.

The Data

Data Sources

Personalized Content :- This table will be provided by the client that will have content based on each Topic and tag specified by the client
Client/Survey Data: This data is being populated by NetWorthy clients on signup/survey.
Roles: Managed by Admin. Currently has 3 roles Client, Coach and Admin.
Client Coach Relation: This data is being populated internally when the client chooses a coach from the list of available coaches.
Goal: This data is being populated internally when the client enters his goal on the NetWorthy portal.
Coach Comments: This data is being populated when the coach enters a comment regarding a particular client's goal.
Tag Topic Relation: The table is being populated by the admin.

Data Structures and Relationships

Data naming

ClientProfile(CLI_*)

Attribute Name	Data Type	Initial Value	Description
CLI_CLIENT_ID(Primary Key)	Varchar	NOT NULL	Client's User ID
CLI_EMAIL	Varchar	NOT NULL	Client's Email Address
CLI_FIRST_NAME	Varchar	NOT NULL	Client's First Name
CLI_LAST_NAME	Varchar	NOT NULL	Client's Last Name
CLI_DOB	Date ('YYYY_MM-DD')	NOT NULL	Client's Date of Birth
CLI_GENDER	Varchar	NOT NULL	Client's Gender
CLI_OCCUPATION	Varchar	NOT NULL	Client's Occupations
CLI_EDUCATION	Varchar	NOT NULL	Client's Highest Level of Education
CLI_UNIVERSITY	Varchar	NOT NULL	University Attended
CLI_LOCATION	Varchar	NOT NULL	Client's Location

CLI_FINANCIAL_LEVEL	Int	NOT NULL	Level of Financial Literacy Range (1 - 10)
CLI_LERNING_METHOD	Varchar	NOT NULL	Preferred Learning Method. Ex – books, videos etc
CLI_INCOME	Int	NOT NULL	Client's Annual Income
CLI_DEBT	Int	NOT NULL	Clients Total Debt
CLI_GENERAL	Varchar	NOT NULL	Why does the client want to improve their finances
CLI_PROFILE_STATUS	Boolean	NOT NULL	Is Client's profile Active/Inactive

CoachProfile (COA_*)

Attribute Name	Data Type	Initial Value	Description
COA_COACH_ID(Primary Key)	Varchar	NOT NULL	Coach's Email Address
COA_EMAIL	Varchar	NOT NULL	Coach's Email Address
COA_FIRST_NAME	Varchar	NOT NULL	Coach's First Name

COA_LAST_NAME	Varchar	NOT NULL	Coach's Last Name
COA_DOB	Date ('YYYY_MM-DD')	NOT NULL	Coach's Date of Birth
COA_GENDER	Varchar	NOT NULL	Coach's Gender
COA_OCCUPATION	Varchar	NOT NULL	Coach's Occupations
COA_EDUCATION	Varchar	NOT NULL	Coach's Highest Level of Education
COA_UNIVERSITY	Varchar	NOT NULL	University Attended
COA_LOCATION	Varchar	NOT NULL	Coach's Location
COA_RESUME	Blob	NOT NULL	Coach's Resume
COA_LOR_1	Blob	NOT NULL	Coach's Letter of Reference 1
COA_LOR_2	Blob	NOT NULL	Coach's Letter of Reference 2
COA_CREDS	Varchar	NOT NULL	Coach's credentials visible to the user directly
COA_PROFILE_STATUS	Boolean	NOT NULL	Is Coach's profile Active/Inactive

UserLoginPassword (ULP)

Attribute Name	Data Type	Initial Value	Description
ULP_USERID(Primary Key)	Varchar	NOT NULL	Client or Coach user ID
ULP_PASSWORD	Varchar	NOT NULL	Client or coach Password
ULP_ROLE	Varchar	NOT NULL	Coach/Client
ULP_IS_VERIFIED	Boolean	NOT NULL	If the user is verified

ClientCoachRelation (CCR)

Attribute Name	Data Type	Initial Value	Description
CCR_COACH_ID (Primary Key)	Varchar	NOT NULL	Coach's User ID
CCR_CLIENT_ID (Secondary Key)	Varchar	NOT_NULL	Client's user IDs

Roles (ROL)

Attribute Name	Data Type	Initial Value	Description
ROL_NAME (Primary Key)	Varchar	NOT NULL	Client/Coach/Admin

Goal (GOL)

Attribute Name	Data Type	Initial Value	Description
GOL_ID (Primary Key)	int	NOT NULL	Goal ID
GOL_CLIENT (Secondary Key)	varchar	NOT_NULL	Client ID
GOL_STATUS	Varchar	NOT_NULL	The current status of the goal.
GOL_REVIEW_COACH	Varchar	NOT_NULL	Coach ID of the coach who will approve the completion of Goal.
GOL_TITLE	Varchar	NOT_NULL	Goal title
GOL_DESCRIPTION	Varchar	NOT_NULL	Goal description.

GOL_S	Varchar	NOT_NULL	Goal ‘Specific’.
GOL_M	Varchar	NOT_NULL	Goal ‘Measurable’
GOA_A	Varchar	NOT_NULL	Goal ’Attainable’
GOA_R	Varchar	NOT_NULL	Goal ‘Relevant’
GOA_T	Varchar	NOT_NULL	Goal ’Time-based’
GOA_TAGS	Varchar[]	NOT NULL	Relevant Tags Associated with the goal

Coach comments (COM)

Attribute Name	Data Type	Initial Value	Description
COM_ID (Primary Key)	int	NOT_NULL	Comment’s ID
COM_GOAL_TITLE	Varchar	NOT_NULL	Goal Title
COM_CLIENT_ID (Secondary Key)	Varchar	NOT_NULL	Client ID

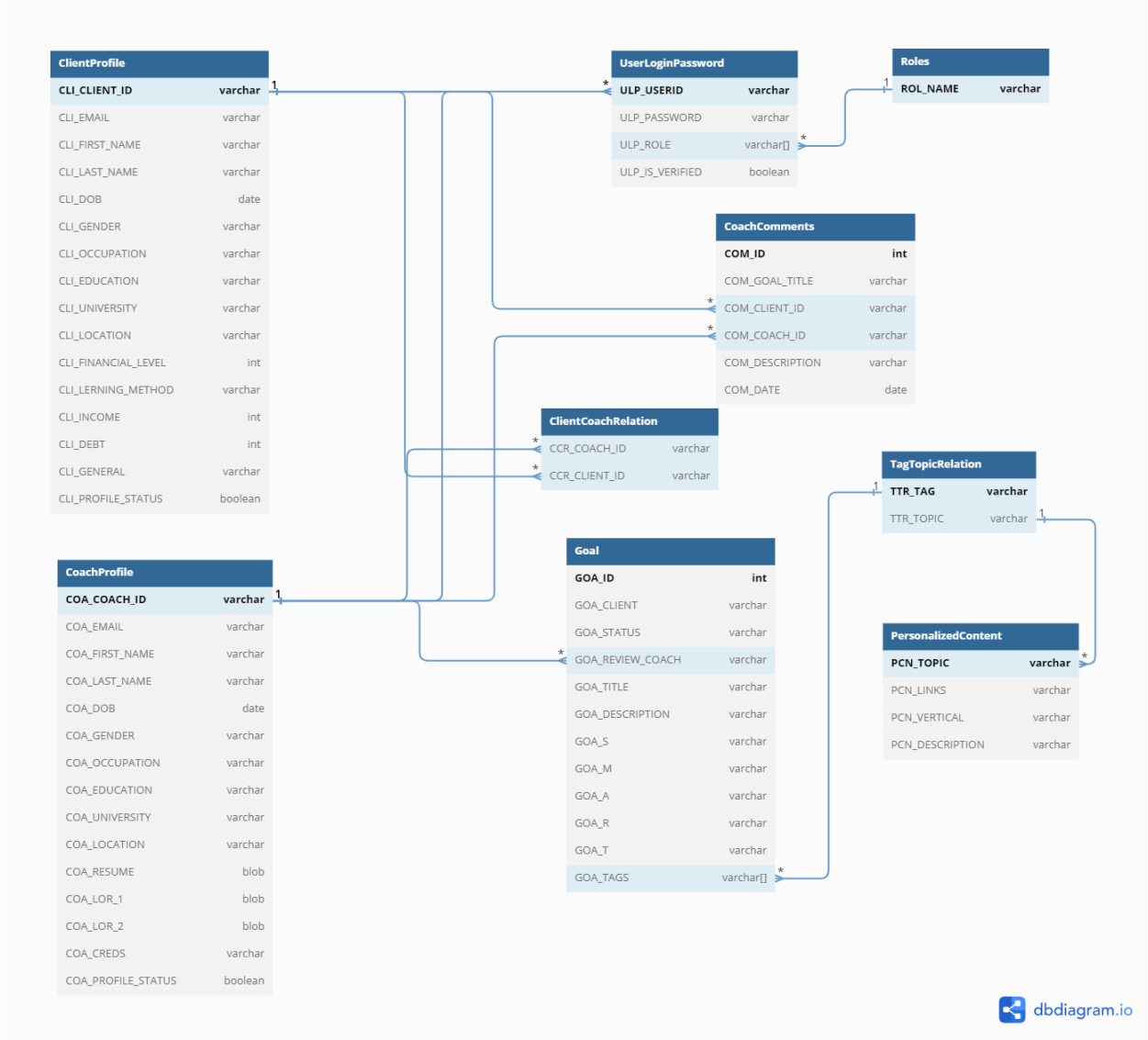
COM_COACH_ID (Secondary Key)	Varchar	NOT_NULL	Coach ID
COM_DESCRIPTION	varchar	NULL	Comments left by the coach for the client
COM_DATE	Date	NOT NULL	TImestamp of the meeting

Personalized Content(PCN)

Attribute Name	Data Type	Initial Value	Description
PCN_TOPIC (Primary Key)	int	NOT NULL	Goal ID
PCN_LINKS	varchar	NOT_NULL	URL related to the content
PCN_VERTICAL	Varchar	NOT NULL	Business Function
PCN_DESCRIPTION	Varchar	NOT NULL	Description

Tag Topic Relation (TTR)

Attribute Name	Data Type	Initial Value	Description
TTR_TAG (Primary Key)	varchar	NOT NULL	Tag Name
TTR_TOPIC	varchar	NOT_NULL	Topic Name



Columns vs. Child Tables

MongoDB supports storing collections of data, which allows you to store multiple elements under one single attribute. We have used this for a few table designs like Tags and Roles.

Cross Reference Tables

We have 2 major profile tables that are Client Profile (ClientProfile) and Coach Profile (CoachProfile). According to our requirements one client can have only 1 coach at a given time. There are different APIs that will be required to fetch all clients for a given coach and client's corresponding Goals, for this we have created a cross reference table that has Coach to Client Mapping (ClientCoachRelation).

The clientID and coachID is referenced across different tables, for example UserLoginPassword, Goals etc.

Design for the future

Our current design incorporates 3 different roles.
ROLE_CLIENT
ROLE_COACH
ROLE_ADMIN.
Each role is independent currently but we have the design in such a way that for future use cases a COACH can also be an ADMIN.
We have defined a Primary Key for each table and a Secondary Key for some. These keys can be used to create indexes on the table in future when we have faster data access as the amount of data grows.

The System

System Description

The project Networthy is a Web Application with a FrontEnd component (ReactJS) interacting with Backend API's (Java). The backend performs CRUD operation on the database (MongoDB) to support all the functionalities of the application. There are three different types of users on the Net Worthy portal: Admin, Client and Coach.

All three users can login the application where the frontEnd calls the Backend API and grants access if the credentials match the database entry. The Client and Coach also need to fill in a survey form upon sign up. The survey fields are different for both these users and therefore we have two different API's for them in the login and signup service.

For the Client User, The login API returns the user's info which will be used to populate the home page of the client. The home page allows a client to create/edit goals which will invoke the API in the client service. The client page will also contain personalized content based on the client's goal tags. This data will be fetched on the frontend using a client service API. The client can also select/delete/change their coach from the list of coaches which will also be a client service API.

The Coach user info will be populated using the coach service API and another API will fetch their client info and goals. The coach can approve an in-review goal using a coach service API. The Coach can also add/view their post meeting comments using another coach service API.

The Admin user can approve a coach using an Admin Service API. The admin can also view/delete any client or coach using another Admin Service API.

System Requirements & Specifications:

1. NetWorthy is a Web Application with minimal third party integrations to Calendly.
2. Components:
    a. Frontend -
        i. Based primarily on React (a JS framework);
        ii. Can be accessed via Web page.
    b. API (Application Programming Interface)
    c. Backend Server -
        i. Based primarily on Java;
        ii. Can be accessed via Local console, Postman API Console.
    d. Database -
        i. MongoDB is being used as a DB.
        ii. Production DB instance maintained via MongoDB Atlas.
3. For version control, we are using Git. This includes the new code deployment and fetch for the latest code from a central repository on Github. For developer / collaborator authentication on Github, we use SSH keys.
4. There is no limitation on the number of API calls per day.

System Diagram