Functional requirements

1. Store user details
2. Store generated CVs
3. Provide multiple templates
4. Add custom templates
5. Allow user to save CV
6. Allow user to update and edit CV at any time
7. Allow user to import CV in sharable format (pdf, image, doc)
8. Suggest relevant skills and achievements to users
9. Allow user to edit provided templates
10. Categorize templates based on purpose - industry and academia
11. Blog posts about CV from experts
12. Identify External Social Links
13. Filter based on category
14. Search based on tags
15. Integration with job search platform (i.e linkedIn)
16. Rating system
17. Like and dislike CVs
18. Suggest CV based on user profile
19. Show user profile and preferences
20. Multiple language support
21. Disability impairment accessibility features based on user type
    - Screen reader for visually impaired person
    - Text to speech for person with dyslexia or visual impairments

Non Functional requirements

1. Store CVs securely
2. Protect user data and privacy
3. Simple and user friendly user interface
4. Minimal loading time
5. Scalable database
6. Caching
7. User friendly error messages
8. Responsive design
9. Fault tolerance

10. Disability impairment accessibility
11. Handle a high number of requests at a certain time

Proposed solution

1. Technology Stack Description:
    - We are using MERN (Mongodb, Express, React, Nodejs) stack for our app.
    - Redux state management
    - Github actions for CI pipeline
    - Deployment in Azure cloud service
    - Backend - Node js, Express
    - Frontend = React, bootstrap
    - Database - Mongodb
        - Library - Mongoose

2. Architecture:
    a. Client - Server approach
    b. NoSQL database
3. Design:
    a. Intuitive design using react bootstrap
    b. Editable cv form with drag and drop support
    c. Change theme
4. Performance:
    a. Code splitting - improve app performance and reduce load times. (i.e. Webpack)
    b. Caching - Memcached can be used to implement server caching and browser caching
    c. Lazy loading - loads resources only when they are needed
    d. Load testing: Load testing tools like Apache JMeter

5. Security:
    a. Authentication and Authorization - Industry standard authentication protocols like Google OAuth v2.O, JWT and password encryption (i.e. bcrypt)
    b. Security-focused Library - tools like helmet.js to add an extra layer of security
    c. Implement Rate Limiting - Prevent brute force attacks on API endpoints (i.e. express-rate-limit in react)
    d. Implement CORS: Restrict access to API from unauthorized domains and somewhat prevents CSRF

Methods

Authentication and Authorization

There will be a custom Authentication module to allow Users to Create an Account and Sign in to access their CVs. We intend to use Json Web Tokens For authentication keeping it secure and not store session data on servers.

Create, Edit or Delete CVs

RestFul APIs will be used for CRUD operations With MongoDB with Mongoose, Hosted on an Express Server and Node.js Backend. For authorized users there will be a user profile. Every user will have a unique ID. Every Cv will have a creator ID which will store the user or creator Information. For non-authorized Creations and Editing there will be a temporary user table.

Creating and CustomizingTemplates

Templates can be created using blocks or components like react-dnd and react-rnd. User will be able to create blocks that can be dragged, dropped, resized, and more. Then the whole configuration can be stored from methods provided by the libraries that return the current configuration the user has on a template.

Exporting to Pdf

To export the CV into a PDF we intend to use react-pdf library or any library that exports react components and css styles into a pdf.

Categorize Templates and CVs and Filtering

Can be easily done by adding a category field to the cv or template schema. Can be used for filtering to recommend to users. Filtering can be done by Node.js.

Blog Posts from Experts

Can be done by implementing a separate router for CRUD that is only known to admins. The Client end will only be able to read the posts and interact with them by liking.

Identify External Social Links

Can be done by logic and by identifying, icons can be set depending on the social link.

Liking and Rating System

Users can like CVs, One user can only like a CV once, If clicked twice, the CV will be disliked, done by Update operation in Node. A user can also rate CVs out of 5. And The user can access the liked CVs from their profile. Can be done with Node.js Create and Update operations.

Search based on Tags

Fuse.js for fuzzy searching and elastic search for searching across indexed data.

User Profile

User Profile that stores All the preferences and information of the user. Stored in the database, fetched by node and exhibited by react.

Suggest CV based on Profile

Can recommend CVs by filtering according to user details (i.e. interests, education etc.) and recommend the top rated CVs

Multiple Language Support

There will be a button that will toggle between languages.

Screen Reader and Text-to-speech

We intend to use Hugging face API to implement usability features.

Caching

Memcached can be used to cache frequently accessed data or objects in memory, which can significantly improve application performance and reduce the load on the database.

Code Splitting and Lazy Loading

Can be implemented via react.lazy and suspense from react and code splitting can be done using webpack.

Sharding and Scalability

Can be achieved through mongodb cluster sharding.

ERD