When your child needs a hospital, everything matters.

Diversify: A web-based applicant tracking system to promote diversity, equity and inclusion in computational genomics

Lauren Griffith, Peter White, Will Mooney, Adam Herman, Meena Punniaraj, Cameron Lloyd

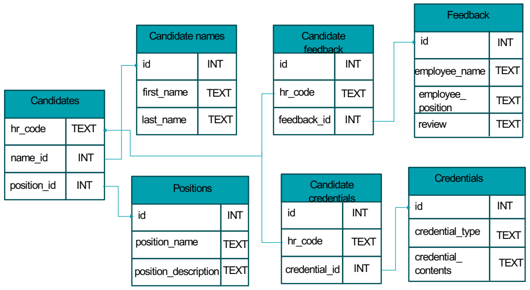
Computational Genomics Group, Institute for Genomic Medicine The Abigail Wexner Research Institute at Nationwide Children's Hospital

Introduction

- Bioinformatics software development is a field that struggles to attract women and candidates from underrepresented groups
- To address this issue, we developed a web application to keep track of diverse candidates who had previously applied for a job in mind for future positions
- As no system was already in place, the project was a full stack development project and required the creation of a database, API and a user interface.

Methods

- Designed tables to hold information for database



Candidates	
hr_code	TEXT
name_id	INT
position_id	INT

Candidate names	
id	INT
first_name	TEXT
last_name	TEXT

Positions	
id	INT
position_name	TEXT
position_description	TEXT

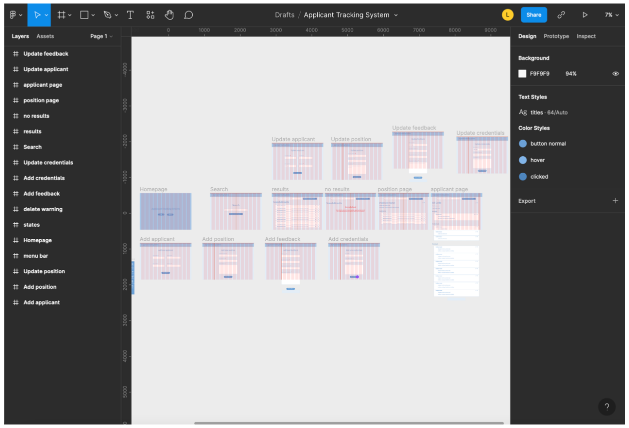
Candidate feedback	
id	INT
hr_code	TEXT
feedback_id	INT

Candidate credentials	
id	INT
hr_code	TEXT
credential_id	INT

Feedback	
id	INT
employee_name	TEXT
employee_position	TEXT
review	TEXT

Credentials	
id	INT
credential_type	TEXT
credential_contents	TEXT

- Designed API endpoints to create, update, view or delete applicant information
- Created UI design and prototype using Figma:



System Testing

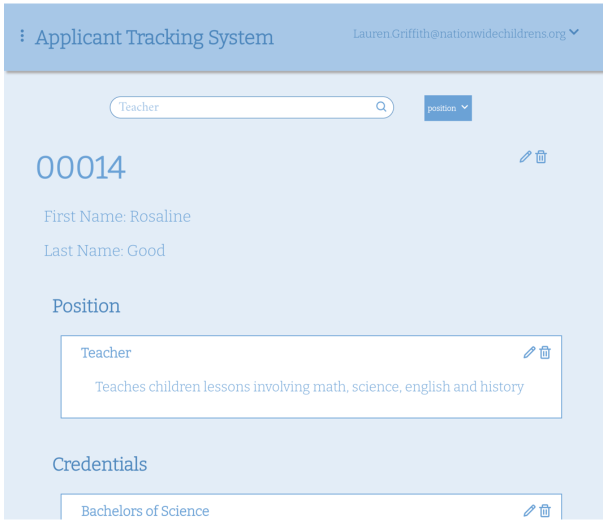
Application was sent to Meena Punniarai, the Clinical Informatics Software Tester, to be reviewed:

Working well:

- Nice color selection
- Simple and neat

Could be improved:

- The search feature could be improved
- A dropdown option for already existing HR Codes and Positions would be beneficial



Applicant Tracking System

Lauren.Griffith@nationwidechildrens.org

Teacher

position

00014

First Name: Rosaline

Last Name: Good

Position

Teacher

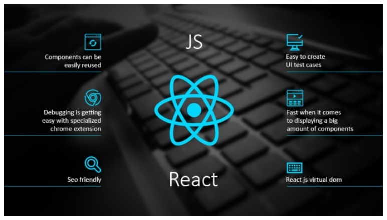
Teaches children lessons involving math, science, english and history

Credentials

Bachelors of Science

Technology used

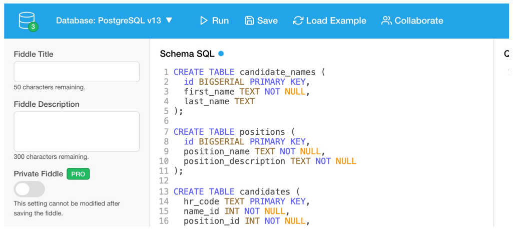
- Deployed using Amazon Web Services (AWS).
- Tables created in AWS using Amazon Relational Database Service (RDS).
- API database connection was adapted to RDS database
- Cognito Authentication added to UI
- API and the UI connected using Axios
- API and UI committed to AWS CodeCommit repositories to be built.



JS
Components can be easily reused
Easy to create UI test cases
Debugging is getting easy with specialized chrome extension
Fast when it comes to displaying a big amount of components
Seo friendly
React js virtual dom
React

Discussion

- Created and tested database using PostgreSQL 13 in DB Fiddle



Database: PostgreSQL v13 ▼ ▷ Run Save Load Example Collaborate

Fiddle Title

50 characters remaining.

Fiddle Description

300 characters remaining.

Private Fiddle PRO

This setting cannot be modified after saving the fiddle.

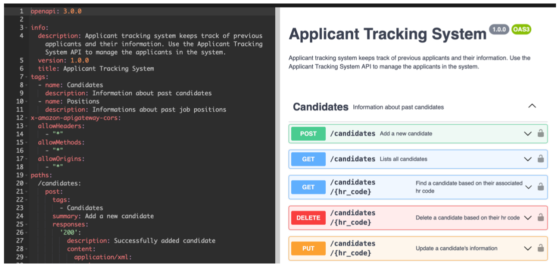
Schema SQL

```sql
CREATE TABLE candidate_names (
  id BIGSERIAL PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name TEXT
);

CREATE TABLE positions (
  id BIGSERIAL PRIMARY KEY,
  position_name TEXT NOT NULL,
  position_description TEXT NOT NULL
);

CREATE TABLE candidates (
  hr_code TEXT PRIMARY KEY,
  name_id INT NOT NULL,
  position_id INT NOT NULL,
```

- Implemented API with OpenAPI 3.0.0, "a standard, language-agnostic interface to RESTful APIs"[1] using Swagger Editor.



```yaml
openapi: 3.0.0

info:
  description: Applicant tracking system keeps track of previous
    applicants and their information. Use the Applicant Tracking
    System API to manage the applicants in the system.
  version: 1.0.0
  title: Applicant Tracking System
tags:
  - name: Candidates
    description: Information about past candidates
  - name: Positions
    description: Informations about past job positions
x-amazon-apigateway-cors:
  allowHeaders:
    - "*"
  allowMethods:
    - "*"
  allowOrigins:
    - "*"
paths:
  /candidates:
    post:
      tags:
        - Candidates
      summary: Add a new candidate
      responses:
        '200':
          description: Successfully added candidate
          content:
            application/xml:
```

Applicant Tracking System 1.0.0 OAS3

Applicant tracking system keeps track of previous applicants and their information. Use the Applicant Tracking System API to manage the applicants in the system.

Candidates Information about past candidates

POST /candidates Add a new candidate

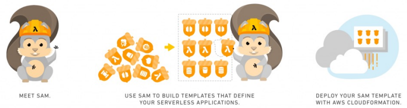
GET /candidates Lists all candidates

GET /candidates/{hr_code} Find a candidate based on their associated hr code

DELETE /candidates/{hr_code} Delete a candidate based on their hr code

PUT /candidates/{hr_code} Update a candidate's information

- Lambda handler functions were written to implement API endpoints using AWS (SAM).

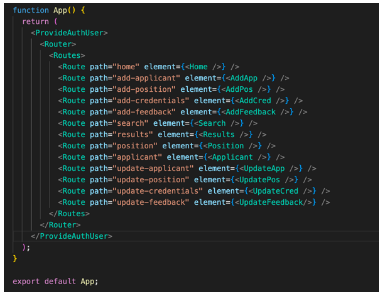


MEET SAM.

USE SAM TO BUILD TEMPLATES THAT DEFINE YOUR SERVERLESS APPLICATIONS.

DEPLOY YOUR SAM TEMPLATE WITH AWS CLOUDFORMATION.

- Connected API to a database in Docker and tested using Postman
- Implemented UI using React and TypeScript
- Used React Router to navigate to each page by connecting a URL path to a component
- Styling done using CSS and transitions added to menus using CSSTransitions.
- Tested UI using Cypress, an end-to-end testing framework that quickly can perform user interactions for front end testing [2].



```jsx
function App() {
  return (
    <ProvideAuthUser>
      <Router>
        <Routes>
          <Route path="home" element={<Home />} />
          <Route path="add-applicant" element={<AddApp />} />
          <Route path="add-position" element={<AddPos />} />
          <Route path="add-credentials" element={<AddCred />} />
          <Route path="add-feedback" element={<AddFeedback />} />
          <Route path="search" element={<Search />} />
          <Route path="results" element={<Results />} />
          <Route path="position" element={<Position />} />
          <Route path="applicant" element={<Applicant />} />
          <Route path="update-applicant" element={<UpdateApp />} />
          <Route path="update-position" element={<UpdatePos />} />
          <Route path="update-credentials" element={<UpdateCred />} />
          <Route path="update-feedback" element={<UpdateFeedback/>} />
        </Routes>
      </Router>
    </ProvideAuthUser>
  );
}

export default App;
```