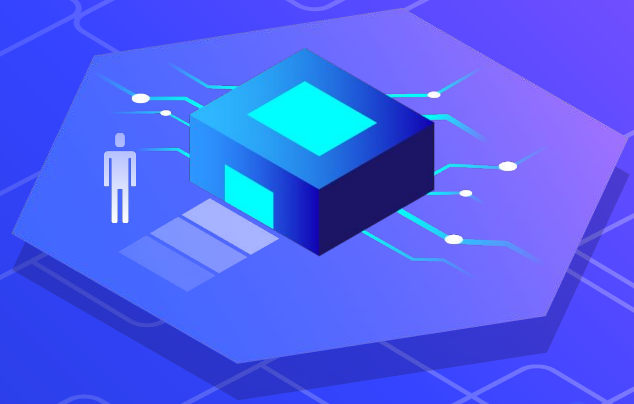
Team 12: United for Literacy

Partner Introduction

United for Literacy
Littératie Ensemble

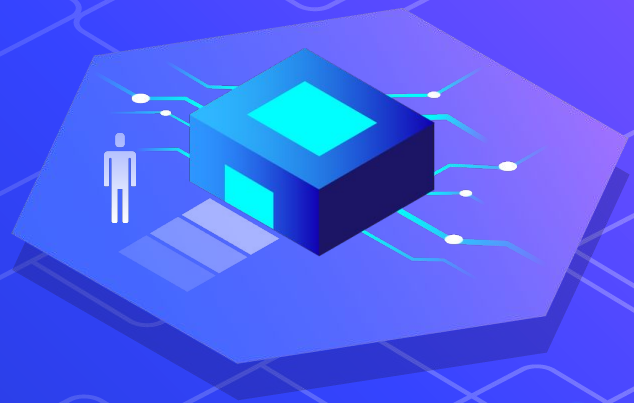
- National charitable organization aiming to increase literacy across Canada

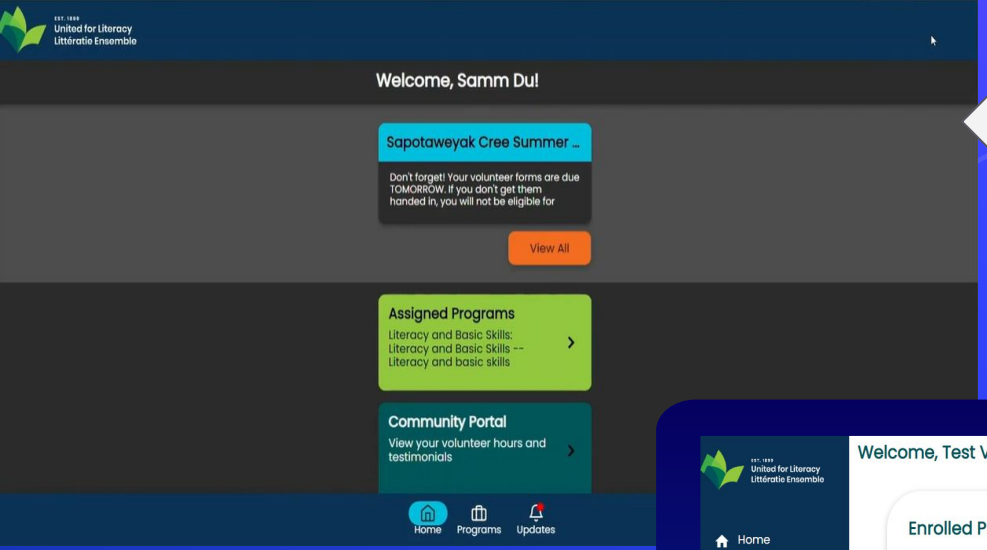


Problem & Solution

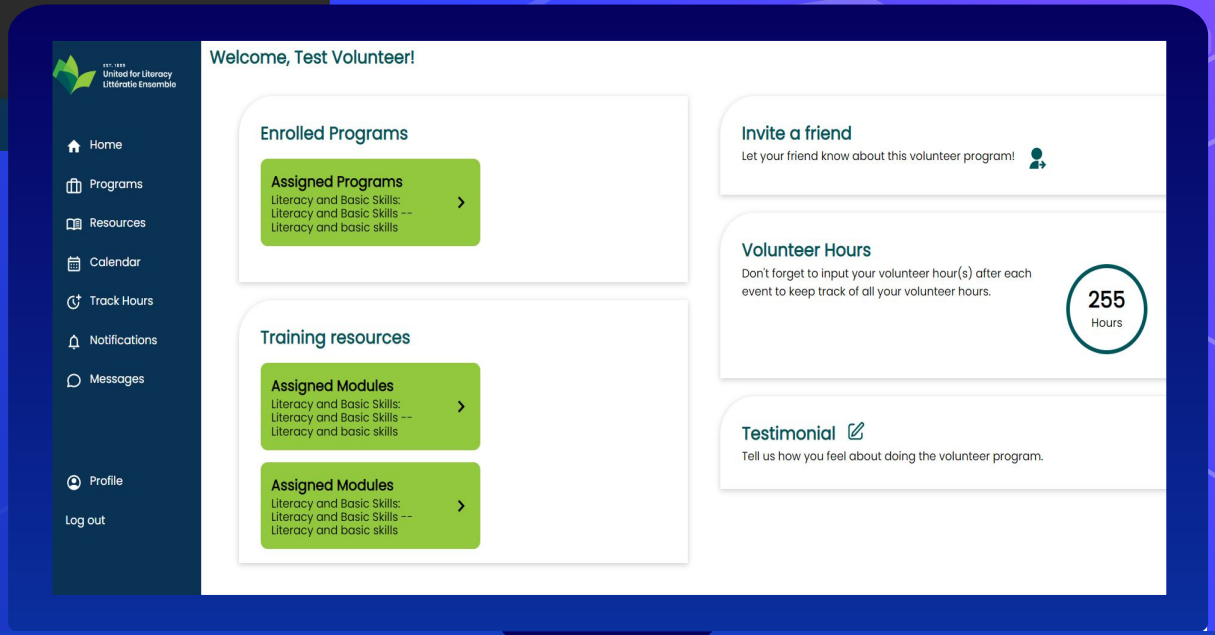
Problem:
No centralized platform for volunteers and staff to access resources and communicate

Solution:
A web app that serves as centralized platform with salesforce integration





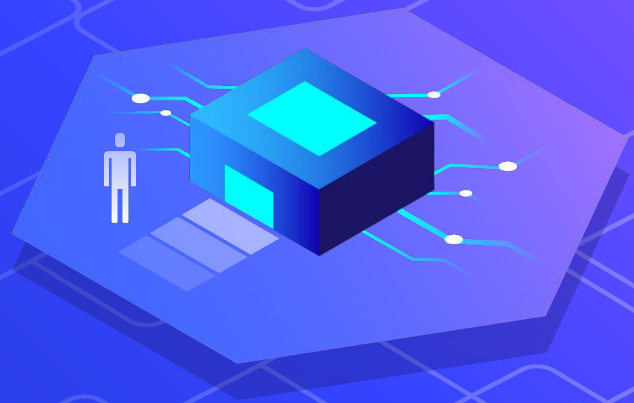
Previous project



Improved UI

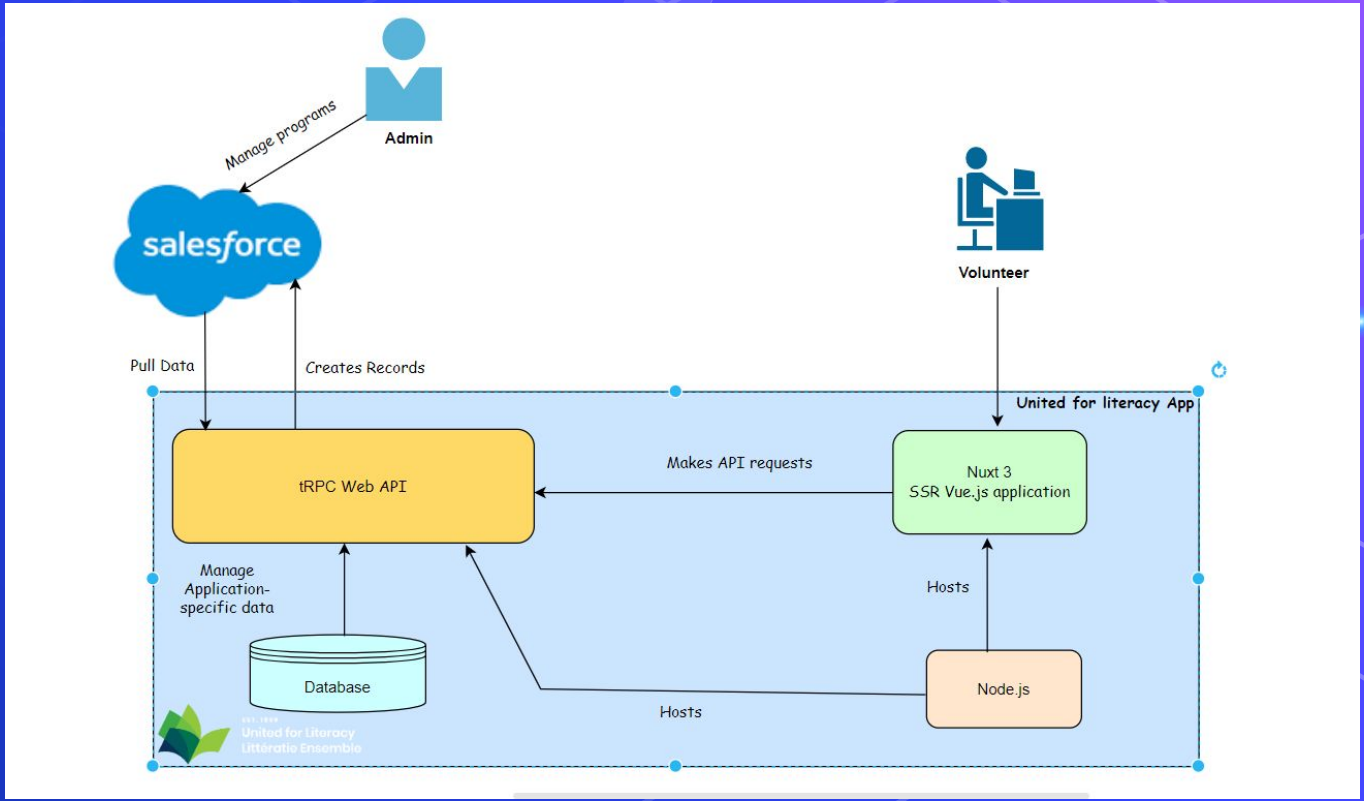
Demo

A live demonstration of our application



Architecture & Technical Discussion

Web Application Architecture



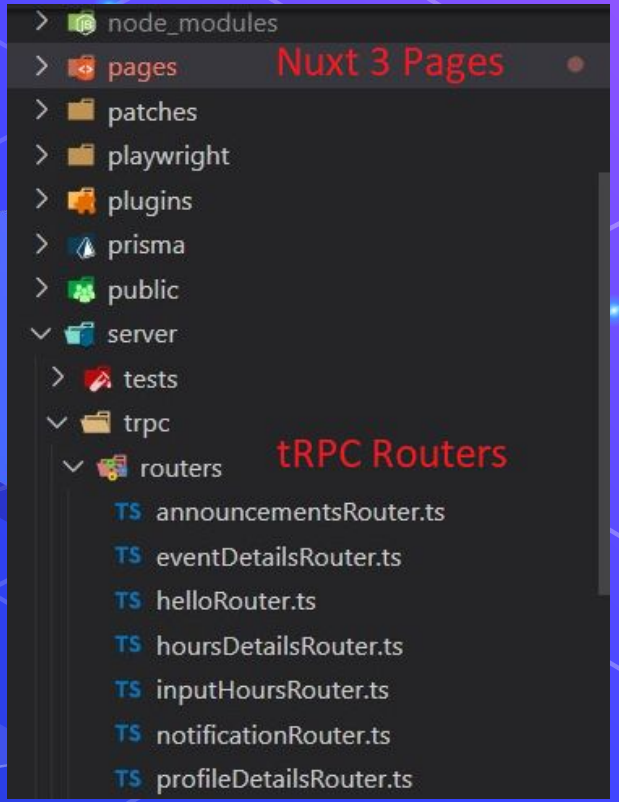
Architecture & Technical Discussion

Why Nuxt 3?

- Building applications easily and quickly
- Modular architecture
- Ability to create static websites

Why tRPC?

- Automatic type-safety
- Allows type sharing between client and server
- Easily create and consume APIs over HTTP



Architecture & Technical Discussion

Connect HTML pages to server's APIs using tRPC

server > trpc > routers > TS testimonialRouter.ts > ...

```
1   import { z } from 'zod'
2   import { auth, api } from '../salesforce'
3   import { createRouter, TRPCError } from '../createRouter'
4
5
6   export const createTestimonialRouter = createRouter().mutation('createTestimonial', {
7     input: z.object({
8       userID: z.string(),
9       programID: z.string(),
10      role: z.string(),
11      otherRole: z.string(),
12      topics: z.string(),
13      story: z.string(),
14      date: z.string()
15    }),
16    async resolve({ input }) {
```

Create the router 'createTestimonial'

server\trpc\routers\testimonialRouter.ts

Architecture & Technical Discussion

Create your HTML page easily with Nuxt 3

```
                <div class="mt-6 mb-4">
                    <textarea id="testimonial-story" v-model="story" placeholder="Write
                    your testimonial here" rows="4" class="w-full p-2" ></textarea>
                </div>

                <!-- <FCButton text="Submit"/> -->
                <button @click="submit">Submit</button>

            </form>
        </div>
    </div>
</template>

<script>
export default {
  data() {
    return {
      testimonial_date: '',
      programID: '',
      role: '',
      otherRole: '',
      topic: '',
      story: ''
    };
  },
  methods: {
    async submit(){
        const client = useClient()
        await client.mutation('createTestimonial', {
            programID: this.programID,
            role: this.role,
            otherRole: this.otherRole,
            topics: this.topic,
            story: this.story,
            date: (new Date(this.testimonial_date)).toISOString()
```

Create your HTML page

AND

Use the router

pages\testimonial.vue

Architecture & Technical Discussion

Nuxt 3 is cool!

Creating your client side API automatically using the page name here we get: “[ServerName]/resources”

You can create components and reuse them easily.
You can reuse the pages as well

Architecture & Technical Discussion

Salesforce API integration

Fetching data:

Create SOQL Select query using SalesForce API names

The query is encoded into a URI to be used in the endpoint

```
async query(statement: string, authInstance: SFAuth): Promise<Response> {
  if (!authInstance.token) {
    await authInstance.getBearerToken(this)
  }

  const path =
    `${this.baseUrl}/services/data/${this.version}/query?q=` +
    encodeURIComponent(statement)

  const opts = {
    headers: { Authorization: `Bearer ${authInstance.token}` }
  }

  let response = await fetch(path, opts)
```

Fetching data API

server\trpc\salesforce.ts

Architecture & Technical Discussion

Salesforce API integration

Creating records:

Creating a new record using SalesForce API names

Required fields are encoded into a URI to be used in the endpoint

```
async createRecord(
  objectName: string,
  fields: Record<string, any>,
  authInstance: SFAuth
): Promise<Response> {
  if (!authInstance.token) {
    await authInstance.getBearerToken(this)
  }

  const path = `${this.baseUrl}/services/data/${this.version}/sobjects/${objectName}`

  const opts = {
    method: 'POST',
    headers: {
      Authorization: `Bearer ${authInstance.token}`,
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(fields)
  }

  let response = await fetch(path, opts)
```

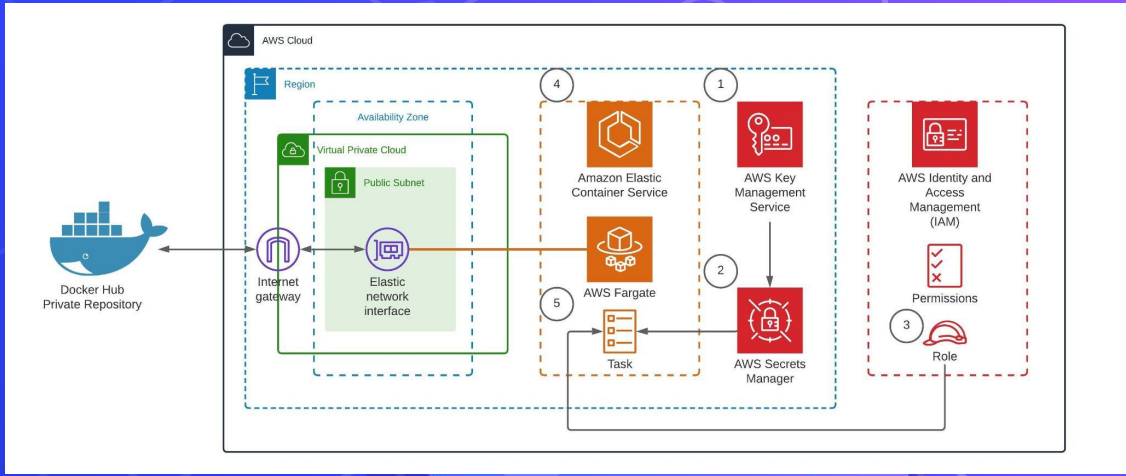
Create a Record API

server\trpc\salesforce.ts

Deployment Process

Why Docker?

- Portability
- Consistency
- Scalability



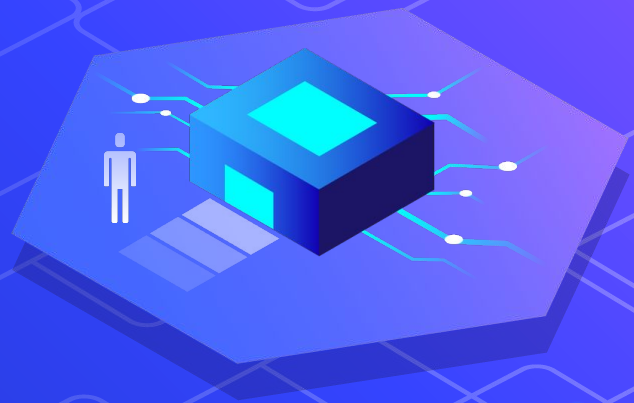
Process

- Create EC2 instance
- Write Dockerfile
- Build the Docker image
- Push image to an AWS registry
- Pull image onto the instance
- Run Docker container on the instance
- Configure Load Balancer
- Deploy!

Key Learnings

- Exposed to operating in a professional software development environment
- Use and collaboration of new developmental tools such as Docker, Nuxt, and Salesforce
- Create a consistent code style convention → Helps future teams
- Improve teamwork as opposed to working on isolated tasks

Next Steps & Handoff



Next Steps & Handoff

The following is an overview what the previous team has completed

Existing User Stories

Home page

- Announcement preview - group it's for, preview of content
- Click on "View All" - take user to "Update" page
- Assigned Programs - take user to "Programs" page
- Community Portal - take user to salesforce login, view volunteer hours and testimonials

Program page

- See all programs user is registered, divided into today and upcoming, clicking on each program card takes user to detail page of that program, including date, address, description, and shared file link to onedrive

Update page

- View all announcements for the user
- Unread red badge disappears after reading
- Refresh button fetches new announcements if any

Environment Variables

The following environment variables also need to be set in the environment.

```
# Production Specific Environment Variables
HOST="0.0.0.0"
NODE_ENV="production"
NPM_CONFIG_PRODUCTIOn="false"

# Replace w/ Production URL
DATABASE_URL="postgresql://root:password@localhost:5433/root?schema=public"

# Fill in these values according to Production Salesforce sandbox
SF_BASE_URL="https://frontiercollege--group467.sandbox.my.salesforce.com"
SF_CLIENT_ID="<SalesForce Client ID>"
SF_CLIENT_SECRET="<SalesForce Client Secret>"
SF_USERNAME="<SalesForce Username>"
SF_PASSWORD="<SalesForce Password>"
```

Prerequisites

- Node.js 16.x
  - Recommend fnm for Node.js version management
- pnpm
- Docker
- VS Code for the best experience
  - Don't forget to install the recommended VS Code extensions when prompted
  - Enable "Takeover Mode" for Vue typechecking - Instructions

United for Literacy Web App

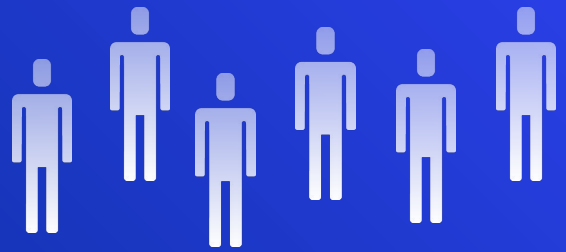
This project is a continuation of a previous student team.

New Features Requested

- Invite friend or colleague to volunteer function
- Accounts for different users (staff, volunteer, admin)
- Volunteer groups where they can communicate/message
- Volunteers can msg supervisor (In-app messaging)
- Integration w/ External Calendars
- Possible to iframe existing SF community pages or website pages? (we can show the pages we're hoping to include on a call)
- Integrate volunteer hour tracking
- Integrate testimonial entry
- Brighten UI colour scheme

Next Steps & Handoff

- Detailed instructions within README.md
- Updated deployment steps
- Handoff email with a demo video included



Individual Contributions

Corinne Lee Slew: UI redesign, homepage, backend and frontend for volunteer hours and testimonial.

Heng-Kuan(Mcgill) Chen: UI/UX design, backend and frontend volunteer hours page, backend profile page, frontend notifications page redesign. Helped writing documents and the deliverables. Point of contact to our partner.

Sari Hammad: Helped with UI/UX design, created the frontend and backend for the calendar and messages pages, helped with deployment and helped with documentation.

Meet Patel: Front-end Vue pages for programs, profile page. Design changes to the UI and pages developed by other members. Wrote documentation for the deliverables and assignments.

Ibrahim Bess: front-end resources page, helped with backend deployment and helped with documentation and deliverables.

Fatimeh Hassan: Worked on frontend and backend for login page and the connected homepage, helped with UI design, worked a lot on written aspects for all the deliverables and assignments, as well as documentation.