REPORT

명지대학교
MYONGJI UNIVERSITY

캡스톤 디자인2

최종 보고서

학번 : 60205204

이름 : 김태진

소속 : 5조

담당교수 : 박현희 교수님

<목차>

1. 프로젝트 선정 동기

커플들은 자신들의 사랑이 운명적인지 아닌지를 궁금해합니다. 해외 매체를 통해서도 커플이 되거나 부부가 되었는데 나중에 알고 보니 과거에 만난 적이 있었다는 기사를 심심치 않게 읽어볼 수 있었습니다. 그래서 저희는 발전한 디지털 기술을 활용하여 커플들이 이전에 만났던 장소와 시간에 대한 기록을 확인할 수 있지 않을까 하는 궁금증으로 해당 주제를 선정하게 되었습니다.

사용자의 google timeline 데이터를 받아와 시간, 날짜, 위치(위도와 경도)로 필터링한 후 두 데이터를 비교해 Google Map에 어떤 경로로 움직였고 어떤 지점에서 만났는지를 시각적으로 표시해주는 웹 서비스를 제공한다는 목표를 갖고 프로젝트를 시작하였습니다.

Google Timeline 데이터를 수집하고, 데이터에 기록되어 있는 날짜/시간/위치(위도, 경도)를 비교, 분석하여 서로의 교차되는 지점을 특정할 수 있을 것이라고 판단했습니다.

생소한 데이터이고, 새로운 분야로의 도전이라 재밌을 것이라고 생각하여 해당 주제를 선정하게 되었습니다.

가) 프로젝트 목적 (Project Objectives):

① 만남 여부 확인:

Google Timeline 데이터를 기반으로, 두 사용자가 이전에 같은 장소에서 같은 시간대에 있었는지를 확인합니다.

② 자주 만나는 장소 식별:

만남 기록이 있는 경우, 어떤 지역이나 장소에서 자주 만났는지를 식별하고 시각적으로 제공합니다.

③ 기억의 활성화:

커플들이 함께한 특별한 순간들을 다시 경험하고 공유함으로써, 그들 간의 추억을 새롭게 활성화시키는 것이 목적입니다.

④ 프로젝트 기술적인 도전:

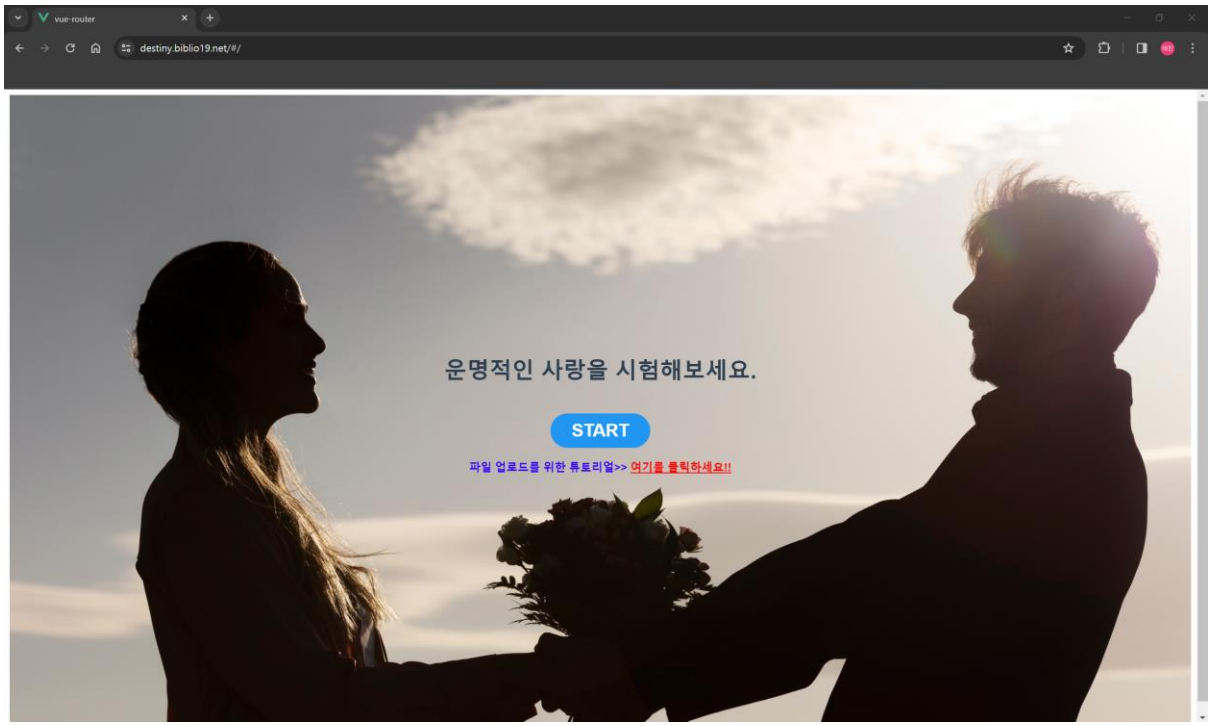
Google Timeline 데이터의 추출, 처리, 지도에 표시하는 등의 기술적인 도전에 대응하며, 실제 데이터를 다루면서 발생하는 문제를 해결하는 것도 목표 중 하나입니다.

나) 개발환경

- Flask
- Google Maps Timeline (API)
- Vue.js
- Python
- Visual Studio Code
- JupyterHub

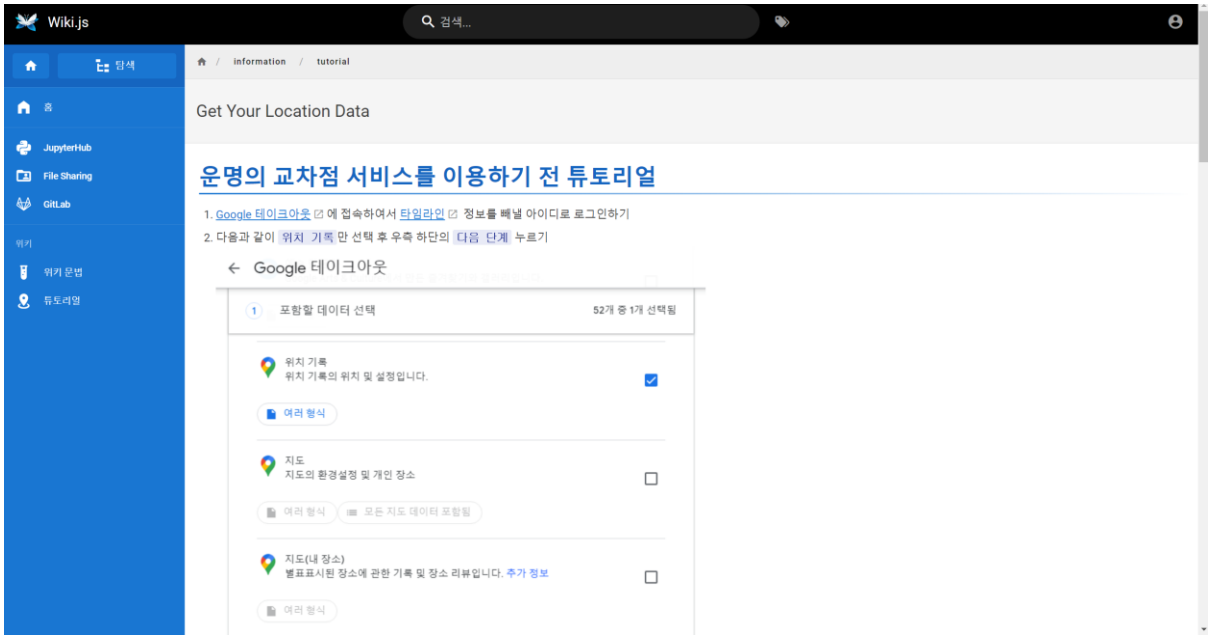
2. 프로젝트 결과

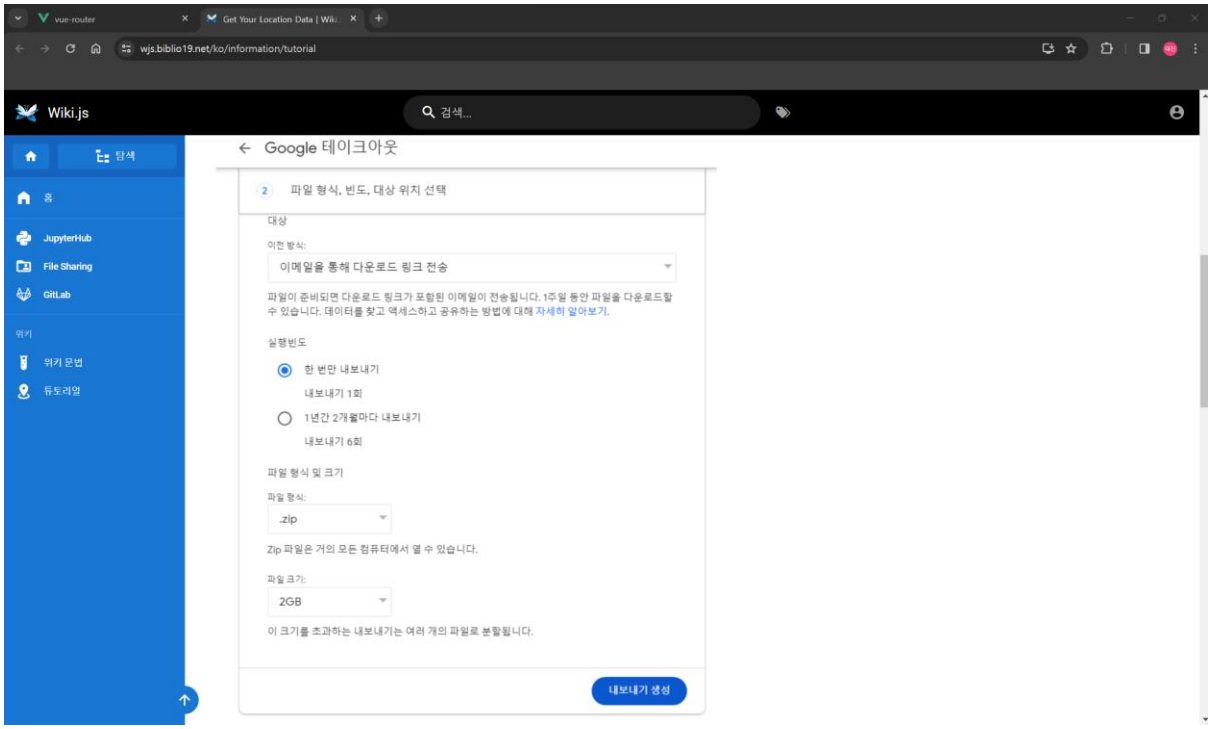
<Home page>



홈페이지, 웹 페이지에서 사용방법을 알려주는 튜토리얼 페이지로 가는 링크를 제공하였습니다. (사용자가 알아볼 수 있게 강조)

<튜토리얼 페이지>



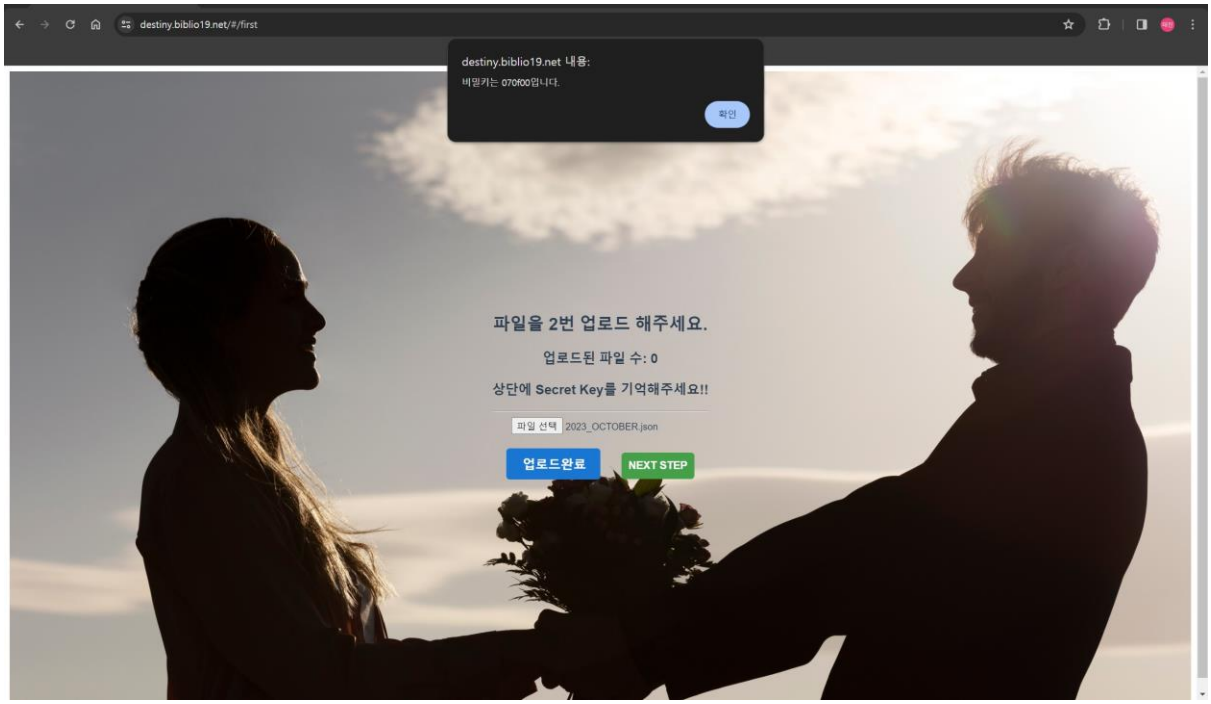


튜토리얼 페이지입니다.

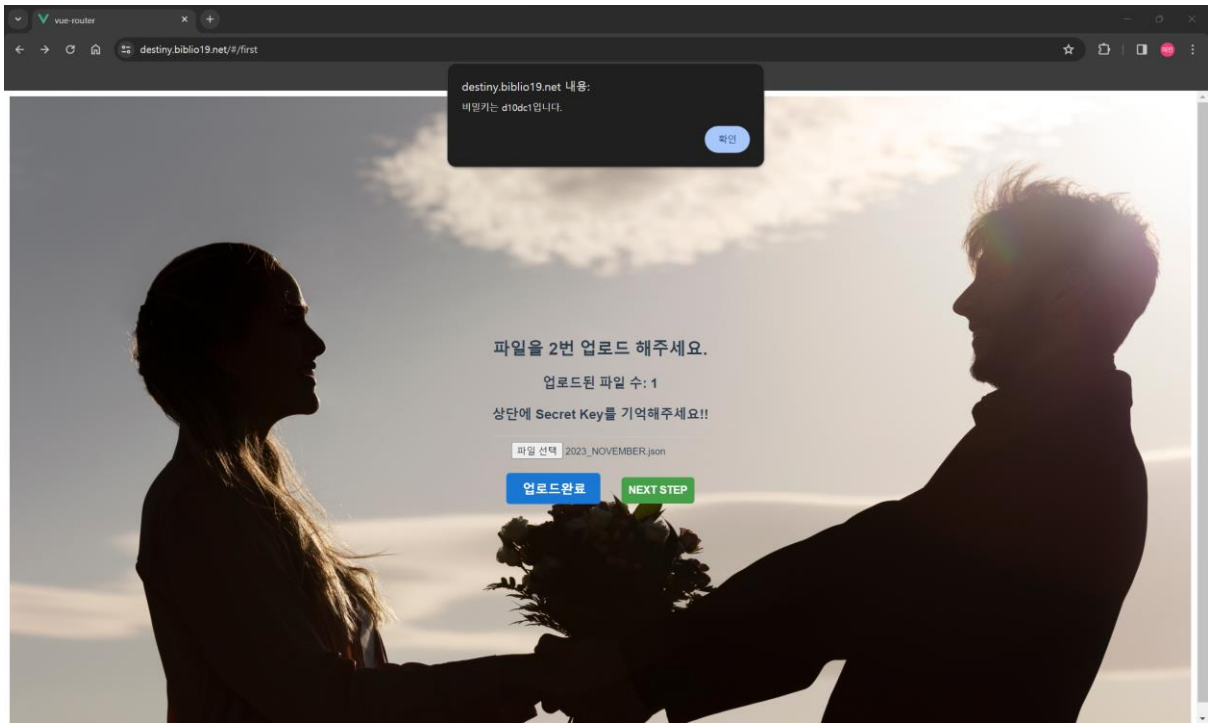
사용할 Json File 을 Export 하는 방법을 자세하게 설명하고 있습니다. DanLab 의 대학원생 분들께 데이터 지원을 부탁드리기 위하여 제공해 드렸는데 문제없이 잘 진행하셨습니다.

데이터 지원: DanLab 대학원생분들 (정말 감사합니다.)

<파일 업로드 페이지>

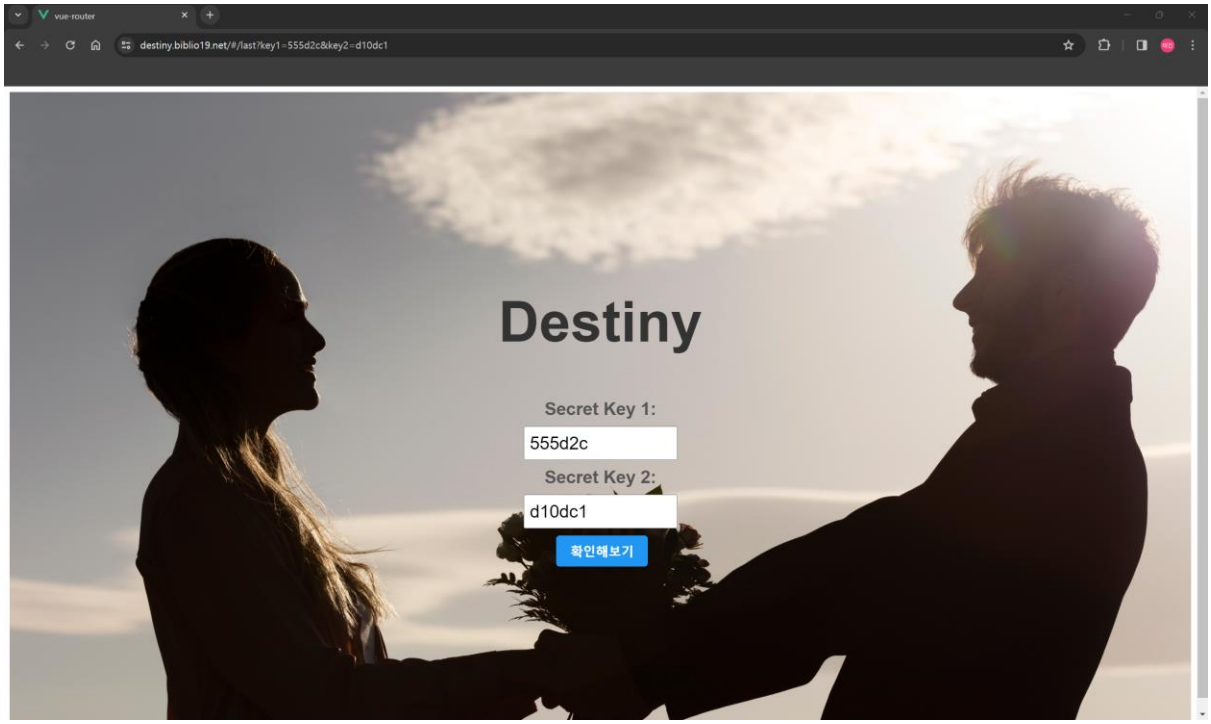


다운 받은 Json File 을 업로드 할 수 있는 페이지입니다. Json File 을 업로드하면 상단에 Secret Key 가 제공됩니다.



하나의 파일을 업로드 한 뒤 카운트가 올라가고 다음 파일을 업로드 할 수 있습니다. 다음 파일을 업로드하면 위에 Secret Key 가 제공됩니다.

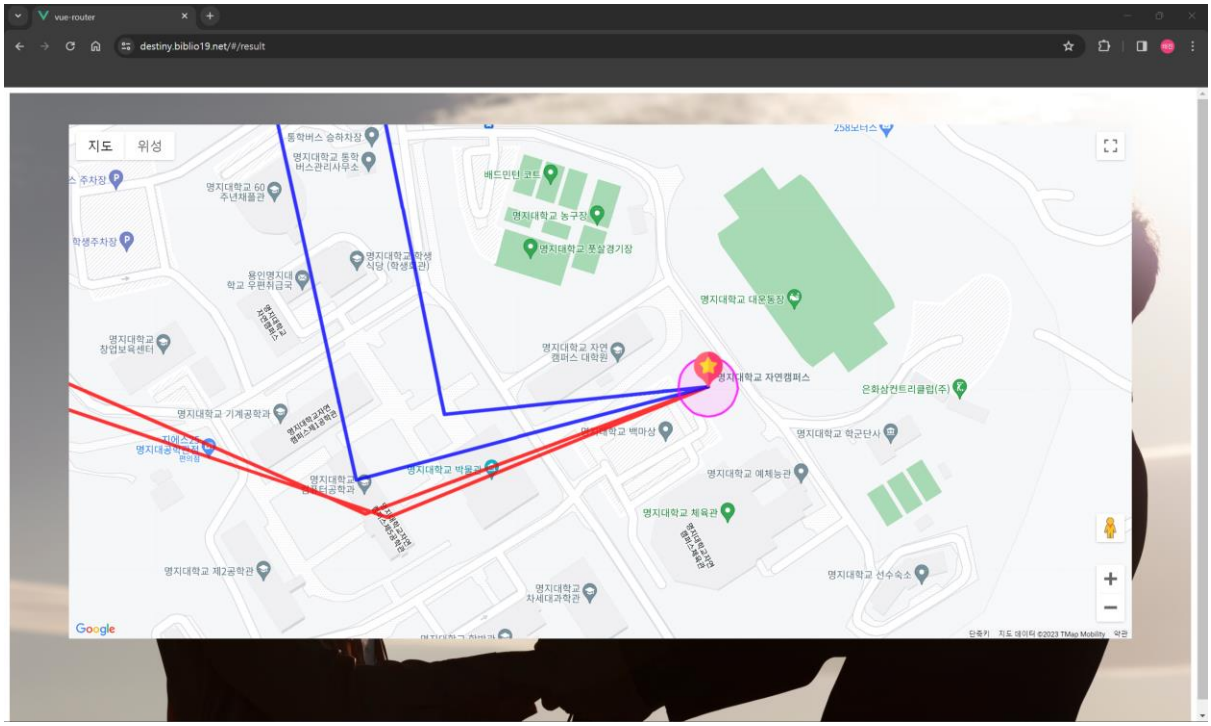
<Secret Key 를 입력하는 페이지>



해당 페이지에서는 전 페이지에서 알려준 Secret Key 를 기억할 필요 없이 자동으로 입력되고 확인해보기 버튼을 누르게 되면, Google Map 이 나오는 페이지로 넘어가게 됩니다.

원래는 Secret Key 없이 해당 파일을 직접 업로드하는 형식이었으나 파일을 업로드 할 때 파일의 이름이 같다면 파일들이 덮어씌워지는 문제가 발생하여서 Backend 에서 파일을 받을 때 파일이름을 변경한 뒤(Secret Key)로 해당 값을 Frontend 로 보내고 이후에 해당 파일을 찾을 때 Secret Key 로 해당 데이터를 찾는 방식으로 변경하여 문제를 해결하였습니다.

<지도 페이지>



이 페이지가 핵심 페이지로 만났던 시간을 기준으로 오차 시간을 +-1 시간까지로 설정하였고, 원의 크기는 사람이 사람이라고 인식할 수 있는 범위인 반지름 20m 로 지정했습니다.

등급	거리	사생활 침해 정도	형태계수
1	10m 미만	사람의 얼굴 표정까지 관찰이 가능한 정도	> 3.18E-03
2	10~18m	사람의 신체적 구분이 뚜렷한 정도	3.18E-03 ~ 9.95E-04
3	18~26m	사람의 신체적 구분을 판단할 정도	9.95E-04 ~ 4.79E-04
4	26~34m	사람의 행위를 명확히 관찰할 수 있는 정도	4.79E-04 ~ 2.80E-04
5	34~42m	사람의 행위를 관찰할 수 있는 정도	2.80E-04 ~ 1.84E-04
6	42~50m	사람의 행위를 판단할 수 있는 정도	1.84E-04 ~ 1.30E-04

<사람 인식범위 참고자료>

출처: https://legalengine.co.kr/cases/6WikyTJW42kEsQztL0BEDw

3. 프로젝트 내 나의 역할

: 저는 프로젝트에서 프론트엔드 관련된 전반적인 것들을 담당했습니다.

먼저 Json File 을 받아올 형식을 설정하고 페이지를 구성했습니다.

페이지는 Home Page, File upload Page, Secret Key 입력 Page, Google Map Page 로 구성했습니다.

페이지 전환은 Vue Router 을 사용하여 원활하게 구현하도록 하였고, Vuex 를 사용하여 각 component 간의 데이터를 전달하였습니다.

웹페이지의 필요한 구성요소들과 그의 디자인을 담당했습니다.

아래는 간단한 코드와 설명입니다.

```vue
<!-- eslint-disable vue/multi-word-component-names -->
<template>
  <div>
    <h1>운명적인 사랑을 시험해보세요.</h1>
    <button class="love-button" @click="goToFirstPage">Start</button>
    <p style="color: #2f00ff; font-weight:bolder ;">파일 업로드를 위한 튜토리얼>> <a href="https://wjs.biblio19.net/ko/information/tutorial" target="_blank" style="color:
  </div>
</template>

<script>
export default {
  methods: {
    goToFirstPage() {
      // 페이지 이동을 수행하는 JavaScript 코드
      window.location.href = '#/first';
    }
  }
};
</script>

<style>
.love-button {
  background-color: #2196F3;
  color: #fff;
  border: none;
  padding: 10px 30px;
  border-radius: 30px;
  cursor: pointer;
  font-size: 25px;
  font-weight: bold;
  text-transform: uppercase;
  transition: background-color 0.3s ease;
  margin-top: 20px;
}
a {
  color: #FFEB3B;
}

a:hover {
  color: #FFC107;
}
```

홈페이지 -> 튜토리얼 링크와 다음 페이지로 넘어가는 버튼 구성

```vue
<template>
  <div class="container">
    <div>
      <h2>파일을 2번 업로드 해주세요.</h2>
      <h3>업로드된 파일 수: {{ uploadCount }}</h3>
      <h3>상단에 Secret Key를 기억해주세요!!</h3>
      <hr />
      <label>
        <input type="file" accept="application/json" @change="handleFileUpload($event)" />
      </label>
      <br />
      <button @click="submitFile" class="upload-button">
        <i class="fas fa-upload"></i> 업로드완료
      </button>
      <button @click="goToFinalRouter" class="final-router-button">
        next Step
      </button>
    </div>
  </div>
</template>

<script>
import axios from 'axios';

const backend_url = process.env.VUE_APP_BACKEND_URL;
let upload_url1 = `${backend_url}/upload_file`;

let key1 = 'Secret Key 1', key2 = 'Secret Key 2';

export default {
  data() {
    return {
      file: '',
      uploadCount: 0, // 업로드된 파일 수를 저장하는 변수
    };
  },
  methods: {
    handleFileUpload(event) {
      this.file = event.target.files[0];
    },
    submitFile() {
      let formData = new FormData();
```

파일 업로드 페이지 --> 정해진 형식으로 Json file 을 Backend 의 URL 로 전송

```vue
  </div>
</template>

<script>
import axios from 'axios';
import { useRoute } from 'vue-router';

const API_URL = process.env.VUE_APP_BACKEND_URL;
let RANDOM_TOKEN = 'analyze';

export default {
  name: 'Last-End',

  data() {
    const route = useRoute();

    const key1 = route.query.key1 ? route.query.key1 : '';
    const key2 = route.query.key2 ? route.query.key2 : '';

    return {
      message: key1,
      message2: key2,
    };
  },

  methods: {
    postData() {
      axios.post(`${API_URL}/${RANDOM_TOKEN}`, {
        key1: this.message,
        key2: this.message2,
        },
        {
        responseType: 'json',
        headers: {
                'Content-Type': 'application/json',
                // Fixed CORS error
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': '*',
                'Access-Control-Allow-Headers': ''
            }
        }
      )
```

마지막에 Json File 을 보내면 Backend 에서 Secret Key 를 보내주고 다시 Secret Key 를 입력하면 Backend 에서 교차점에 대한 데이터를 전달해 줍니다.

```vue
<template>
  <div style="width: 90%; height: 90%">
    <!-- <button @click="showMapMethod();">Show Map</button> -->
    <GoogleMap
    v-if="showMap"
    api-key="AIzaSyDGn4q3xBEYl9Hqvn6Beqjk4yZSVGrL1bE"
    :center="center"
    :zoom="19"
    style="width: 100%; height: 90%"
    >
      <Polyline :options="crossPath1" />
      <Polyline :options="crossPath2" />
      <Circle :options="circle" />
      <Marker :options="markerOptions" />
    </GoogleMap>
  </div>
</template>

<script>
import { defineComponent } from "vue";
import { useStore } from 'vuex';
import { GoogleMap, Polyline, Circle, Marker } from "vue3-google-map";

// console.log(result);

export default defineComponent({
  components: { GoogleMap, Polyline, Circle, Marker },

  data() {
    return {
      showMap: true,
    };
  },

  methods: {
    showMapMethod() {
      this.showMap = true;
    },
  },

  setup() {
```

```vue
setup() {

  const store = useStore();
  let result = JSON.parse(JSON.stringify(store.state.data));

  // Refuse direct access to this page
  if (!result) {
    alert('교차점이 없습니다. 다시 시도해 주세요.');
    window.location.href = '#/last';
    location.reload();
  }

  // console.log(result);
  // console.log(JSON.parse(JSON.stringify(result)))
  // console.log(result.crossing_point[0]);

  let data = result[0].data1[0]
  // let data = result[1].data2[0]

  // console.log(result.data1[0])
  // result = result.data1[0];

  let center = data.crossing_point[0];
  // console.log(center);

  let route1 = data.route1;
  let route2 = data.route2;
  // console.log(route1);
  // console.log(route2);

  const circle = {
    center: center,
    radius: 20,
    strokeColor: "#FF00FF",
    strokeOpacity: 0.8,
    strokeWeight: 3,
    fillColor: "#FF00FF",
    fillOpacity: 0.1,
  };

  const customIcon = {
    url: require('../assets/map-icon-50px.png'),
```

Google Map API 를 이용하여 Google Map 을 구현하고 Polyline 2 개와 Polycircle 를 활용하여 데이터의 경로와 교차지점을 시각화 합니다.

구글 마커도 원래 있던 마커가 아니라 다른 마커 디자인으로 수정했습니다.

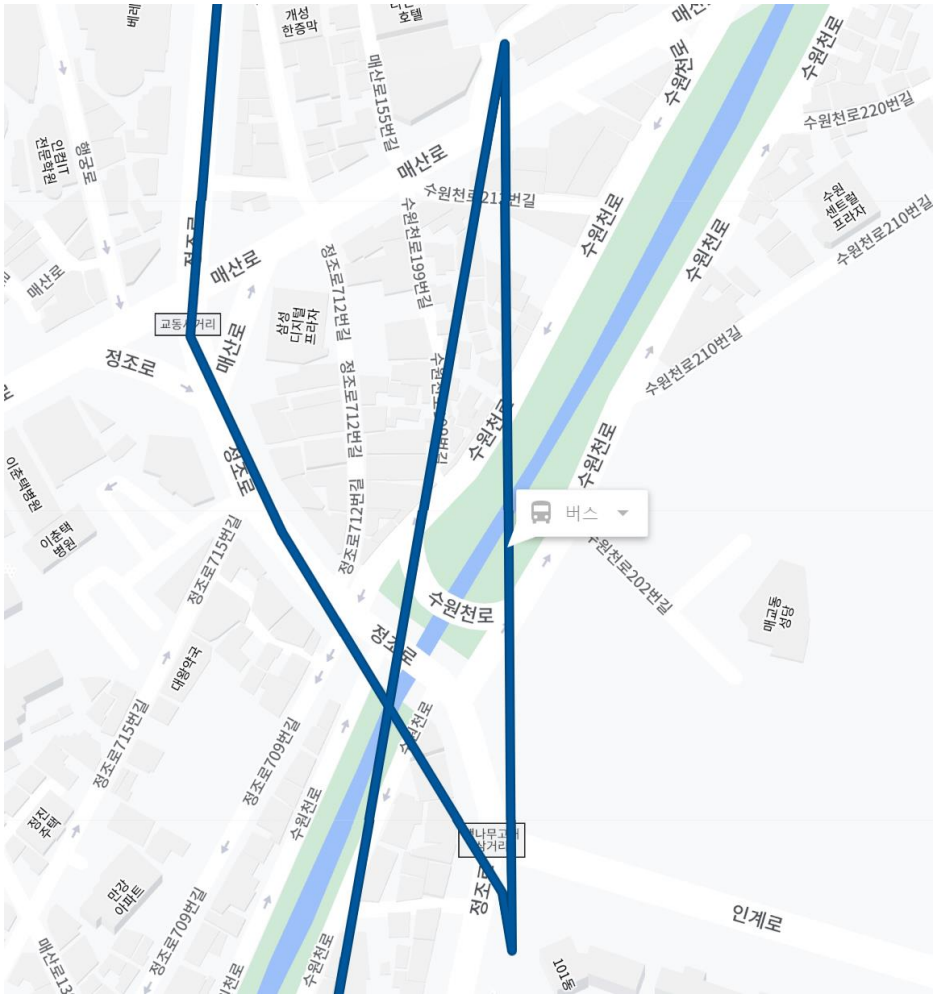
4. 한계 및 아쉬운 점

① 시각적 표현의 한계(데이터의 한계)

지도를 보게 되면 사람의 이동경로가 Google Mas 에 나와있는 길을 따라 이동하지 않고, 마치 길을 뚫고서 이동하는 것처럼 이동경로와 Google Map 의 길이 싱크가 맞지 않는 것을 확인할 수 있습니다. 이는 데이터가 해당하는 길을 따라 이동하는 것처럼 보일 수 있도록 짧은 시간 간격으로 찍혀 있지 않았기 때문에 발생한 문제였습니다.

실제 구글에서 직접 제공하는 Google Map 에 들어가서 내 이동경로를 확인해보면 도보 뿐만 아니라 실제로 버스를 타고 이동할 때도 도로를 따라가지 않고 마치 길을 뚫고서 가는 것처럼 보이는 것을 확인할 수 있습니다.

그리고 평소에 스마트폰의 GPS 기능을 off 하고 생활하는 사용자는 충분한 데이터 수집이 이루어지지 않아서 서비스를 이용하는데 제한이 있을 수 있다는 것 또한 한계점이라고 할 수 있습니다.



<실제 Google 이 직접 제공하는 타임라인 서비스>

② 부족한 컨텐츠

원래는 커플이 만나지 않았을 경우에 대비하여 그러면 만남이 이어진 후에 이동경로나 방문한 장소를 분석하여 근처 명소를 추천해주는 컨텐츠도 초기에는 계획했으나 메인 Flow 에 집중하느라 조금 더 풍부한 컨텐츠를 구현하지 못한 것이 굉장히 아쉽습니다.

또 초기에 카카오톡이나 인스타그램 등에 공유하는 share 버튼도 구현하려고 구상했지만 시간부족으로 아쉽게 구현하지 못했습니다.

③ 데이터 필터링

자주, 오래 같이 있는 장소인 회사, 학교 등과 같은 장소를 필터링하여 불필요한 교차점이 생성되는 것을 방지하려고 계획했으나 메인 작업 우선순위에 밀려 결국 완성하지 못했습니다.

④ 다수의 교차점 표시

여러 개의 경로와 교차점을 조금 더 깔끔하게 표현하고 싶었으나 교차점이 많이 생기는 상황에서의 얽히는 경로와 교차되는 지점을 시각적으로 깔끔하게 표현하지 못하고 데이터 기준으로 가장 최근의 교차되는 지점과 경로만을 표시하게 된 것을 굉장히 아쉽게 생각합니다.

일단은 "만난적이 있었을까" 라는 메인 주제에 맞게 가장 최근 교차점을 지도에 표기하는 것으로 프로젝트를 마무리하게 되었습니다.