Code Review Package

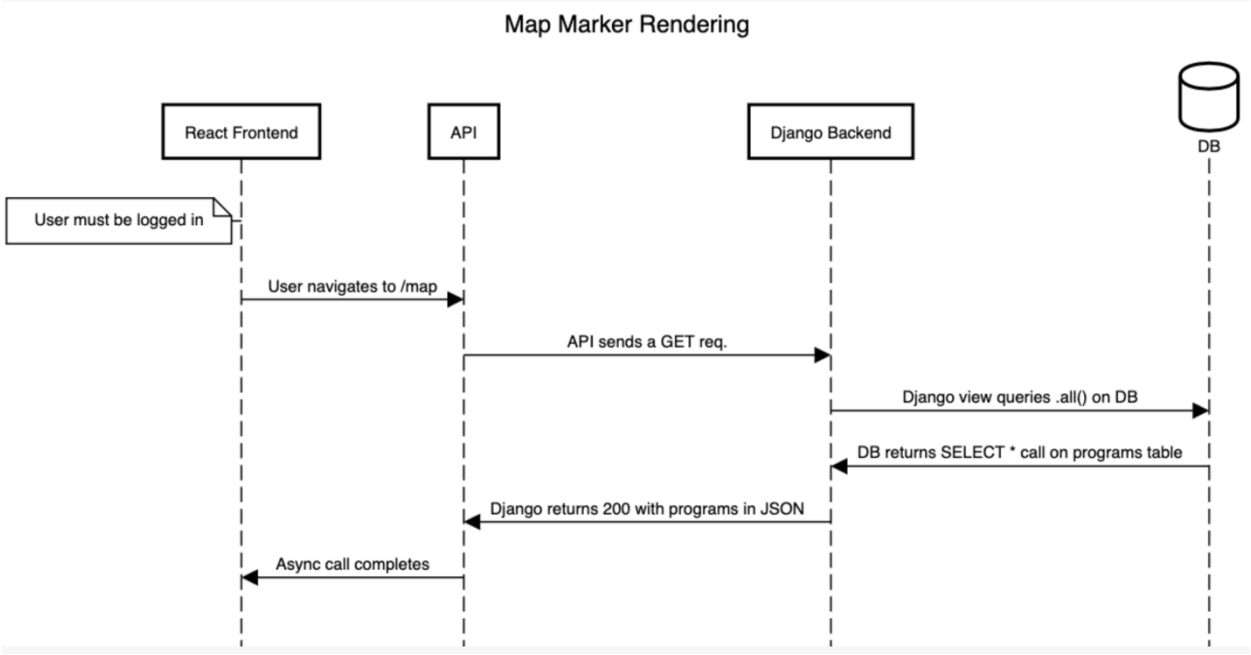
Anchor Abroad – Group 7

Jacob Davey, Ben Mongirdas, Daniel Henricks, Maharshi Patel

Subsystem to review: Rendering locations on the map

Description of the system: As a part of our application, we need to render in the different possible study abroad locations that users could go to. For example, if the user were to be interested in traveling to Europe, some of the possible locations would be Madrid, Paris, Rome, and London. These locations must have a latitude and longitude (to put them on the map somewhere) and a description of the program when clicked on.

Sequence diagram:



Code specification:

This implemented connecting our frontend and backend as well as adding mock data to the map. The primary data structure for the markers on the map can be seen in the Django model:

```python
from django.db import models

class Program(models.Model):
    name = models.CharField(max_length=255, unique=True) # Primary key
    description = models.TextField()
    latitude = models.FloatField()
    longitude = models.FloatField()
```

We have a simple model for now but will add more here in the future. Django packages these back to the frontend in the view:

```python
from django.http import JsonResponse
from django.shortcuts import render
from .models import Program
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
from .serializers import ProgramSerializer

@api_view(['GET']) # What type of HTTP method does this accept?
@permission_classes([IsAuthenticated])
def list_programs(request): # List all the programs.
    programs = Program.objects.all()
    serializer = ProgramSerializer(programs, many=True)
    return JsonResponse(serializer.data, safe=False)
```

And since we are using mock data, we load data into the DB using a script. This can be seen in the management commands folder under programs:

```python
class Command(BaseCommand):
    help = 'Load European city programs into the database'

    def handle(self, *args, **options):
        programs_data = [
            {
                'name': 'Madrid',
                'description': 'The vibrant capital of Spain, known for its rich history, world-class museums like the Prado, and bustling nightlife. Home to the Royal Palace and beautiful Retiro Park.',
                'latitude': 40.4167,
                'longitude': -3.7033,
            }
```

...more programs and logic below.

Source code to review:

Frontend:

App.jsx

```jsx
// App.jsx
import './App.css';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import AuthWrapper from './components/AuthWrapper';
import ProgramDetail from './pages/ProgramDetail';
import MessageDetail from './pages/MessageDetail';
import MapPage from './pages/Map';
import Home from './pages/Home';

function App() {
  return (
    <div className="App">
      <Router>
        <AuthWrapper>
          <Routes>
            <Route path="*" element={<Home />} />
            <Route path="/home" element={<Home />} />
            <Route path="/programs/:id" element={<ProgramDetail />} />
            <Route path="/messages/:id" element={<MessageDetail />} />
            <Route path="/map" element={<MapPage />} />
          </Routes>
        </AuthWrapper>
      </Router>
    </div>
  );
}

export default App;
```

MapPage (step 1 on sequence diagram)

```jsx
// frontend/src/pages/Map.jsx
import { useState, useEffect } from 'react';
import { MapContainer, TileLayer } from 'react-leaflet';
import 'leaflet/dist/leaflet.css';
import { Box } from '@mui/material';
import { MarkerManager } from '../components/marker';
import Sidebar from '../components/sidebar';
import apiService from '../services/api';
```

```jsx
// frontend/src/pages/Map.jsx
const MapPage = ({ mapCenter }) => {
  const [selectedMarker, setSelectedMarker] = useState(null);
  const [sidebarOpen, setSidebarOpen] = useState(false);
  const [markers, setMarkers] = useState([]);

  useEffect(() => {
    apiService
      .getPrograms() // API called here (transition to API layer)
      .then((data) => setMarkers(data)) // Sequence diagram is complete here.
      .catch((error) => console.error('Error fetching markers:', error));
  }, []);

  const handleMarkerClick = (marker) => {
    setSelectedMarker(marker);
    setSidebarOpen(true);
  };

  const handleSidebarClose = () => {
    setSidebarOpen(false);
    setSelectedMarker(null);
  };

  const center = mapCenter || [47.812, 8.4058];

  return (
    <Box sx={{ flexGrow: 1, width: '100%' }}>
      <MapContainer
        center={center}
        zoom={5}
        style={{ height: 'calc(100vh - 64px)', width: '100%' }}
      >
        <TileLayer
          attribution="&copy; OpenStreetMap contributors &copy; CARTO"
          url="https://cartodb-basemaps-
a.global.ssl.fastly.net/light_all/{z}/{x}/{y}{r}.png"
        />
        <MarkerManager markers={markers} onMarkerClick={handleMarkerClick} />
      </MapContainer>

      <Sidebar open={sidebarOpen} onClose={handleSidebarClose}
selectedMarker={selectedMarker} />
    </Box>
  );
};

export default MapPage;
```

MarkerManager:

```jsx
// frontend/src/components/marker.jsx
import { Marker, Popup } from 'react-leaflet';
import markerIconPng from 'leaflet/dist/images/marker-icon.png';
import { Icon } from 'leaflet';

export const MarkerManager = ({ markers, onMarkerClick }) => (
  <>
    {markers.map((marker, idx) => (
      <Marker
        key={idx}
        position={[marker.latitude, marker.longitude]}
        icon={
          new Icon({
            iconUrl: markerIconPng,
            iconSize: [25, 41],
            iconAnchor: [12, 41],
            popupAnchor: [0, -35],
          })
        }
        eventHandlers={{
          mouseover: (e) => e.target.openPopup(),
          mouseout: (e) => e.target.closePopup(),
          popupopen: (e) => {
            if (e.popup && e.popup._container) {
              e.popup._container.style.pointerEvents = 'none';
            }
          },
          click: () => onMarkerClick(marker),
        }}
      >
        <Popup closeButton={false} autoClose={false} closeOnClick={false}>
          <strong>{marker.name}</strong>
          <br />
          {marker.description}
        </Popup>
      </Marker>
    ))}
  </>
);
```

API service:

```
/**
 * API service for handling authentication and other backend communication
 */
// This will need to be changed to the actual URL of the backend server when
deploying.
const API_BASE_URL = 'http://localhost:8000/api';

class ApiService {
  constructor() {
    this.baseURL = API_BASE_URL;
  }

  /**
   * Make HTTP request with proper headers and error handling
   */
  async request(endpoint, options = {}) {
    const url = `${this.baseURL}${endpoint}`;
    const config = {
      headers: {
        'Content-Type': 'application/json',
        ...options.headers,
      },
      credentials: 'include',
      ...options,
    };

    if (config.body && typeof config.body === 'object') {
      config.body = JSON.stringify(config.body);
    }

    try {
      const response = await fetch(url, config);
      const data = await response.json();

      if (!response.ok) {
        if (data.username) {
          throw new Error('A user with that username already exists.');
        }
        throw new Error(data.message || 'Request failed');
      }

      return data;
    } catch (error) {
      console.error('API request failed:', error);
      throw error;
    }
  }
```

```
/**
 * GET requests use this.
 */
async get(endpoint) {
  return this.request(endpoint, { method: 'GET' });
}

/**
 * POST requests use this.
 */
async post(endpoint, data) {
  return this.request(endpoint, {
    method: 'POST',
    body: data,
  });
}

// Our methods for connecting to the backend go here.

/**
 * User signup
 */
async signup(userData) {
  return this.post('/auth/signup/', userData);
}

/**
 * User login
 */
async login(credentials) {
  return this.post('/auth/login/', credentials);
}

/**
 * User logout
 */
async logout() {
  return this.post('/auth/logout/');
}

/**
 * Get current user profile
 */
async getUserProfile() {
  return this.get('/auth/profile/');
}
```

```
// Step 2 of sequence diagram here.
async getPrograms() {
    return this.get('/programs/');
  }
}

const apiService = new ApiService();
export default apiService;
```

Backend:

urls.py (entry point)

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/auth/', include('accounts.urls')),
    path('api/programs/', include('programs.urls')),
]
```

Second urls.py:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.list_programs, name='program_list'),
]
```

View:

```
from django.http import JsonResponse
from django.shortcuts import render
from .models import Program
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated, AllowAny
from .serializers import ProgramSerializer

@api_view(['GET'])
@permission_classes([AllowAny])
def list_programs(request): # List all the programs.
```

```
    programs = Program.objects.all()
    serializer = ProgramSerializer(programs, many=True)
    return JsonResponse(serializer.data, safe=False)
```

Program model:

```
from django.db import models

class Program(models.Model):
    name = models.CharField(max_length=255, unique=True) # Primary key
    description = models.TextField()
    latitude = models.FloatField()
    longitude = models.FloatField()
```

Serializer:

```
from rest_framework import serializers
from .models import Program

class ProgramSerializer(serializers.ModelSerializer):
    class Meta:
        model = Program
        fields = ['id', 'name', 'description', 'latitude', 'longitude']
```

Test suite:

Setup the app using the README. Then run the backend tests using python manage.py test, and the frontend tests by running npm test.