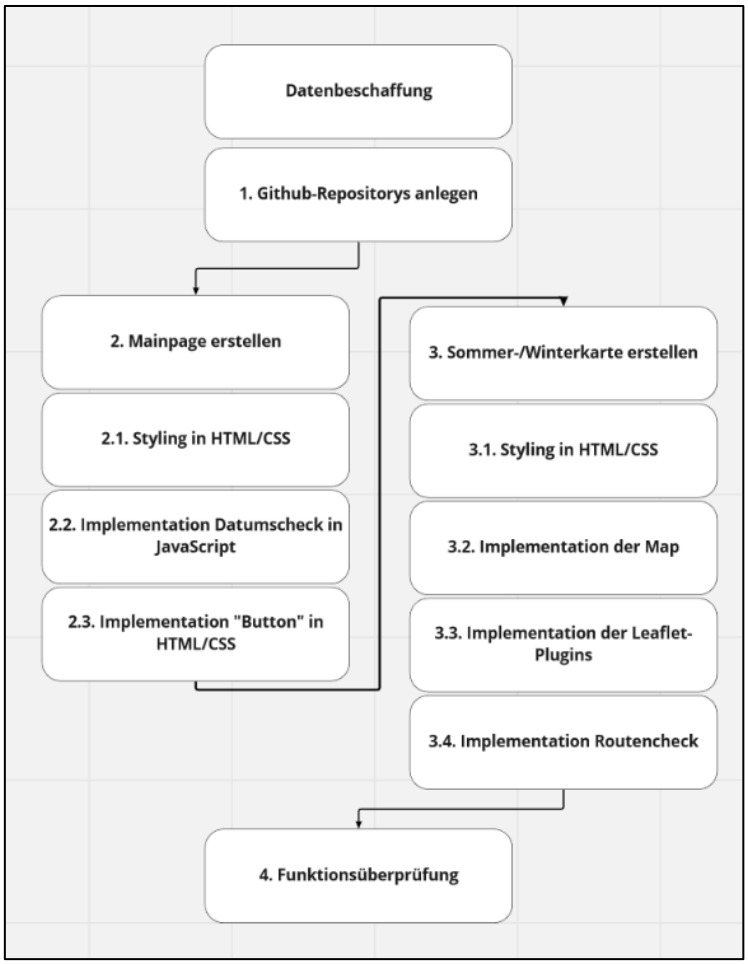
Projektdokumentation Webmapping – Angerer Stefan, Voit Maximilian

1. Projektziel

Erstellen von 3 Webpages. Eine beinhaltet einen Datumscheck und leitet an eine - von zwei - Webpages weiter die entweder eine Sommerkarte (mit einer zufällig ausgewählten Bike-Tour) oder eine Winterkarte (mit einer zufällig ausgewählten Skitour) öffnet. *Optimales Ziel: Routen werden mit Wetter und Sicherheitscheck ausgestattet und nur wenn die Route „gut" durchführbar ist, geladen.*

2. Workflow

Für die Erstellung des Projekts richten wir uns nach folgendem Workflow:



3. Datengrundlage

Als Datengrundlage dienen uns im Projekt folgende Daten:

HTML-Hintergrundbilder:

- Mainpage-Hintergrund: (https://unsplash.com/photos/gdDcR44NKBg)
- Sommerpage-Hintergrund: (https://unsplash.com/photos/Devu2TMGIWU)
- Winterpage-Hintergrund: (https://unsplash.com/photos/oql99W45lTU)

Datengrundlage-Sommerkarte:

- Wetterstationen: (https://www.data.gv.at/katalog/dataset/bb43170b-30fb-48aa-893f-51c60d27056f(
- Radrouten: (https://www.data.gv.at/katalog/dataset/land-tirol_mountainbikeroutenintirol/resource/d5c1a8fe-dca3-4b0f-864d-b6f4a46169bd)

- Almdaten: (https://www.data.gv.at/katalog/dataset/land-tirol_almzentrenintirol/resource/a31ebbde-e9e1-45c8-93c5-ad99cc7663cd)

Datengrundlage-Winterkarte:

- Wald und Wildschutzlayer: (https://data-tiris.opendata.arcgis.com/datasets/tiris::wald-und-wildschutzzonen.geojson)
- Skitourendaten: (https://alpinverlag.at/gps-daten-skitourenfuehrer-tirol/)

Leaflet-Basemaps:

- Sommerkarte: (https://www.data.gv.at/katalog/dataset/land-tirol_elektronischekartetirol)
- Orthofoto: (https://www.data.gv.at/katalog/dataset/land-tirol_elektronischekartetirol)
- Orthofoto + Besch.: (https://www.data.gv.at/katalog/dataset/land-tirol_elektronischekartetirol)

Leaflet-Plugins:

- Leaflet: (https://leafletjs.com/)
- Providers: (https://github.com/leaflet-extras/leaflet-providers)
- Fullscreen: (https://leaflet.github.io/Leaflet.fullscreen/)
- GPX: (https://github.com/mpetazzoni/leaflet-gpx)
- Markercluster: (https://github.com/Leaflet/Leaflet.markercluster)
- MiniMap: (https://github.com/Norkart/Leaflet-MiniMap)

4. Durchführung

In der folgenden Tabelle wird die Durchführung unseres Workflows dokumentiert. Dabei wird stehts eine Beschreibung + passende Screenshots zur Verfügung gestellt.

Arbeitsschritt	Grafik
1. Github-Repositorys erstellen Analog zum Kurs. Es wurden drei Repositorys für drei verschiedene Webseiten erstellt. Die Repositorys sind unter folgenden Links erreichbar:	
2. Mainpage erstellen	

2.1. Styling in HTML/CSS

Grundlage der Mainpage ist die Page des “Neuseeland Beispiels”. Davon wurde jedoch fast der gesamte Inhalt gelöscht und nur die Content-Box behalten. Der Inhalt wurde mit Headline, Ersteller-Info und einleitendem Text versehen. Die Meta Daten sind analog zum “Neuseeland” Beispiel. Der Import des CSS bzw. JS-Files erfolgen unter diesem Block.

Im CSS-File wurde das Hintergrundbild verlinkt und es werden Stile Vorgaben für Headlines, Text und Buttons implementiert.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>A Day A Tour</title>

    <!-- main.css einbauen-->
    <link rel="stylesheet" href="main.css">
    <!-- main.js einbauen-->
    <script defer src="main.js"></script>

    <!-- font awsome einbauen -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css"
        integrity="sha512-KfkfwYDsLkIlwQp6LFnl8zNdLGxu9YAA1QvwINks4PhcElQSvqcyVLLD9aMhXd13uQjoXtEKNosOWaZqXgel0g=="
        crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
  <main class="shadow">
      <!-- generelle informationen einbauen -->
      <article>
          <h1>A DAY A TOUR</h1>
          <h2>Stefan Angerer & Maximilian Voit</h2>
          <p class="blocked">For outdoor people Tirol has simply too many possibilities. With this little
          tool here, we want to help all ski and bike enthusiasts and make the choice for their next
          tour. Divided by season. With all the information you need for outdoor fun.
          </p>
          <!-- find a tour button einbauen -->
          <div class="box-2">
              <div class="btn btn-two">
                  <span onclick="window.location.href=link;">Find a tour</span>

              </div>
            </div>


      </article>
```

2.2. Implementation Datumscheck in JavaScript

Eine Funktionalität des Skripts beruht auf einem Datumscheck, der den User dann auf eine von zwei Map-Pages weiterleitet. Dieser Datumscheck wurde in JS implementiert. Im Prinzip erstellt das Skript beim Laden der Page eine Variable mit dem aktuellen Datum und vergleicht sie mit den, von uns definierten, Saisons Grenzen. Der Code der Implementierung ist im rechten Screenshot ersichtlich und verdeutlicht den simplen Charakter.

Eine Herausforderung im Projekt war die Einbettung dieses Algorithmus in einen Button im HTML-Script. Schlussendlich konnte aber mithilfe eines Span-Objekts und der onclick variable mit value “window.location.href=link” ein “workaround” gefunden werden.

```
winterStart = new Date(1900, 11, 01)
winterEnd = new Date(1900, 03, 31)
today = new Date()

var link

if ((today.getMonth() > winterEnd.getMonth()) && (today.getMonth() < winterStart.getMonth())){
    link = "https://skitourentirol.github.io/sommer/"
} else {
  link = "https://skitourentirol.github.io/skitouren.github.io/"
}
      <!-- find a tour button einbauen -->
      <div class="box-2">
          <div class="btn btn-two">
              <span onclick="window.location.href=link;">Find a tour</span>

          </div>
        </div>
```

2.3. Implementation „Button“ und “Menu” in HTML/CSS

Die Implementation der beiden User-Interacatble Objekte erfolgt im Wesentlichen in CSS. Dabei verlassen wir uns auf Code der frei im Internet zur Verfügung steht.

Die Implementation der beiden "user-available" Objekte erfolgt im Wesentlichen in CSS. Dabei verlassen wir uns auf Code der frei im Internet zur Verfügung steht. Für den Random-Route-Button wurde Code von (https://webdeasy.de/en/top-css-buttons-en/) implementiert und für den Menü Button wurde Code von (https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_dropdown_right) implementiert. Beide Codes wurden noch an das Styling der Mainpage angepasst. Damit die ganze Page einen Runden Eindruck hinterlässt.

```css
/* Styling für Season-Button */
.btn {
  line-height: 50px;
  height: 50px;
  text-align: center;
  width: 150px;
  cursor: pointer;
  font-family: 'Open Sans Condensed', sans-serif;
  text-transform: uppercase;
  font-weight: bolder;
  position: absolute;
  top: 50%;
  left: 50%;
  -ms-transform: translate(-50%, -50%);
  transform: translate(-50%, -50%);

}

.btn-two {
  color: rgb(255, 255, 255);
/* Styling für Dropdown-Menu */
.dropbtn {
  background-color: rgba(255, 255, 255, 0.1);
  font-family: 'Open Sans Condensed', sans-serif;
  text-transform: uppercase;
  font-weight: bolder;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
```

3. Sommerpage erstellen

3.1. Styling in HTML/CSS

Beim Styling mit HTML/CSS wurde im Wesentlichen wie bei der Mainpage vorgegangen. Außer das vor dem Text-Block noch die Map eingefügt wurde. Das Hintergrundbild stammt wieder von unsplash.

```css
/* Stile für Body, Map und Texte */

.map {
    padding: 50px 0px 50px 0px;
}

.informations {
    text-align: right;
    color: white;
    background-color: rgba(0, 0, 0, 0.4);
    box-shadow:
        0 4px 8px 0 rgba(0, 0, 0, 0.7),
        0 6px 20px 0 rgba(0, 0, 0, 0.7);
        padding: 10px 0px 50px 0px;
}

.informations_text{
    color: rgba(255, 255, 255, 1);
    font-size: 19px;
    font-weight: 600;
    padding-bottom: 20px;
    margin: 0 0 0;
    text-align: left
}
```

3.2. Implementation der Map

Als Grundlage wurden, die im Kurs erstellten Repositories zu „aws-tirol“ und „biketirol“ als Templates verwendet. Damit wurden die notwendigen Wetterdaten in die Karte implementiert und die Voraussetzung für das Implementieren von GPX Tracks gesetzt.

Anschließend wurden die Templates zu einer Neuen Karte zusammengeführt und entsprechende Anpassungen vorgenommen.

Die Icons für die Seite wurden angepasst sowie die eGrundkarte Tirol Sommer als Startlayer festgelegt.

Die Almen wurden mit dem Datensatz „Almzentren“ als geoJson hinzugefügt. Der Name der Alm wurde hierbei als Pop-up hinzugefügt. Hier konnte leider nicht mehr Information implementiert werden, da die geoJson Datei keine weitere nützliche Information enthält.

Weiterhin wurden für die Sommerkarte die Biketrails Tirol als geoJson aus OpenData eingebunden. Diese wurden anschließend nach ihrer Schwierigkeit klassifiziert und entsprechend eingefärbt (leicht = grün; blau = mittelschwierig; rot = schwer). Zusätzlich wurden die Biketrails als GPX Dateien eingebunden, um damit den Zufallsgenerator zu entwickeln. Die GPX Daten und das geoJson der Biketrails überschneidet sich und liegt übereinander.

Um für die Touren immer das korrekte Wetter sehen zu können, wurden verschiedene Layer aus dem Repository „aws-tirol“ wiederverwendet und implementiert.

```
// Radrouten Tirol Anzeigen Geojson
async function loadTracks(url) {
    let response = await fetch(url);
    let geojson = await response.json();
    console.log(geojson);
    let overlay = L.featureGroup();
    layerControl.addOverlay(overlay,"Bikerouten");
    overlay.addTo(map);

    L.geoJSON(geojson, {
        style: function(feature) {
            let colors = {
                "schwierig" : "#FF4136",
                "mittelschwierig": "#0074D9",
                "leicht": "#2ECC40",
            };
            return {
                color: `${colors[feature.properties.SCHWIERIGKEITSGRAD]}`,
                weight: 4,
                dashArray: [10, 6]
            }
        }
    }).bindPopup(function (layer) {
        return `
            <h4>${layer.feature.properties.ROUTENNAME} (${layer.feature.properties.ROUTEN_TYP}) </h4>

            von: ${layer.feature.properties.ROUTENSTART}<br>
            nach: ${layer.feature.properties.ROUTENZIEL}<br>
            <p> ${layer.feature.properties.ROUTENBESCHREIBUNG}</p>
            <p><li> StreckenLänge: ${layer.feature.properties.LAENGE_HAUPTROUTE_KM} km</li>
            <li> Fahrzeit: ${layer.feature.properties.FAHRZEIT}</li>
            <li> Hoehenmeter bergauf: ${layer.feature.properties.HM_BERGAUF} m</li>
            <li> Hoehenmeter bergab: ${layer.feature.properties.HM_BERGAB} m</li></p>
        `;

    }).addTo(overlay);
}
loadTracks("https://data-tiris.opendata.arcgis.com/datasets/tiris::radrouten-tirol.geojson");
```

```
// Almen anzeigen
async function loadHuts(url) {
    let response = await fetch(url);
    let geojson = response.json();
    let overlay = L.markerClusterGroup();
    layerControl.addOverlay(overlay,"Almen");
    overlay.addTo(map);

    L.geoJSON(geojson,{
        pointToLayer: function(geoJsonPoint,latlng){
            let popup = `
                <strong>${geoJsonPoint.properties.NAME}</strong>
            `;

            return L.marker(latlng, {
                icon: L.icon({
                    iconUrl: "icons/almen.png",
                    iconAnchor: [16,37],
                    popupAnchor: [0,-37]
                })
            }).bindPopup(popup);
        }
    }).addTo(overlay);
}
loadHuts("https://opendata.arcgis.com/datasets/cd1b86196f2e4f14aeae79269433a499_0.geojson");
```

```
// GPX Track Layer implementieren with random Track
let gpxTrack = new L.GPX(str, {
    async: true,
    marker_options: {
        startIconUrl:"icons/start.png",
        endIconUrl: "icons/finish.png",
        shadowUrl: null,
        iconSize: [32, 37],
        iconAnchor: [16, 37],
    },
    polyline_options: {
        color: "black",
        dashArray:[2, 5],
    },
}).addTo(overlays.gpx);

gpxTrack.on("loaded", function(evt) {
    console.log("Loaded gpx event: ", evt);
    let gpxLayer = evt.target;
    map.fitBounds(gpxLayer.getBounds());
}).addTo(map);
```

3.3. Implementation der Leaflet-Plugins

Folgende Leaflet-Plugins wurden verwendet:

Leaflet fullscreen plugin,
Leaflet GPX plugin
Leaflet Markercluster
Leaflet MiniMap
Leaflet elevation (optional)

Die Almen wurden aufgrund ihrer Anzahl mithilfe eines MarkerClusters konfiguriert.

```
// Layer control ausklappen
layerControl.expand();

// Maßstab control
L.control.scale({
    imperial: false
}).addTo(map);

// Minimap
var osm2 = new L.TileLayer("http://wmts.kartetirol.at/gdi_summer/{z}/{x}/{y}.png",
{minZoom: 6, maxZoom:7, attribution: "https://www.data.gv.at/katalog/dataset/land-tirol_elektronischekartetirol"});
var miniMap = new L.Control.MiniMap(osm2).addTo(map);

// Fullscreen control
L.control.fullscreen().addTo(map);
```

```
// Almen anzeigen
async function loadHuts(url) {
    let response = await fetch(url);
    let geojson = await response.json();
    let overlay = L.markerClusterGroup();
```

```
// Optional (falls Daten mit Höhe vorliegen)
// let elevationControl = L.control.elevation({
//     time: false,
//     elevationDiv: "#profile",
//     theme: "bike-tirol",
//     height: 200,

// }).addTo(map);
// gpxTrack.on("addline", function(evt){
//     elevationControl.addData(evt.line);

// });
```

3.4. Implementation des Zufallsgenerators

Um den GPX Track beim Laden zufällig anzuzeigen, sind mehrere Voraussetzungen notwendig. Zunächst mussten die Grunddaten entsprechend aufbereitet werden. Die GPX Daten zu den Biketrails lagen bereits in einem Ordner, waren allerdings nicht durchgängig durchnummeriert. Um die Datensätze aufsteigend durchzunummerieren, wurde ein entsprechendes Python Skript mit dieser Funktion geschrieben.

Nun wurde mit Math.floor(Math.random()*); eine zufällige Zahl generiert, die zwischen null und der eingegebenen Variablen liegt. Anschließend wurde die erzeugte Zahl in einen String umgewandelt damit sie dann dem Pfad übergeben werden konnte, sodass das die zufällig ausgewählte GPX Datei geladen wird.

```
import os
path = r"C:\VertiGIS\Uni\Webmapping\
    skitouren.github.io\data\Skitouren\Routen\GPX_Only"
os.chdir(path)
files = os.listdir(path)


for index, file in enumerate(files):
    os.rename(os.path.join(path, file),
    os.path.join(path, ''.join([str(index), '.gpx'])))
```

```
// Generate Random Number
let randomNumber = Math.floor(Math.random() * 760);
let strRandom = randomNumber.toString();

// Generate path with RandomNumber
const path = "./data/Radtouren/"
const endPath = ".gpx"
const str1 = path.concat(strRandom)
let str = str1 + endPath
```

Der hier erstellte String wird bei der Implementation des GPX Tracks für den Dateipfad angegeben.	
4. Winterpage erstellen	
4.1. Styling in HTML/CSS … Analog zum Sommerbeispiel. Hintergrundbild von Unsplash. Leicht veränderte Datenlage. Ansonsten alles gleich.	

4.2. Implementation der Map

Die Karte wurde synchron zur Sommerkarte entwickelt, wodurch der Aufbau größtenteils der gleiche ist, wie bei der Sommerkarte. Der Startlayer ist entsprechend der Jahreszeit die eGrundkarte Tirol Winter.

Die individuellen Änderungen sind auf die spezifische Jahreszeit zurückzuführen. Statt den Biketrails wurden Skitouren als GPX Track eingebunden. Weiterhin wurden die jeweiligen Icons an Thema und Jahreszeit angepasst.

Als zusätzlichen Layer, wurde für die Winterkarte, die Wild- & Waldschutzzonen implementiert.

```
// Wildschutzzonen
async function loadZones(url) {
    let response = await fetch(url);
    let geojson = await response.json();
    // console.log(geojson);
    let overlay = L.featureGroup();
    layerControl.addOverlay(overlay, "Wald- und Wildschutzzonen");
    overlay.addTo(map);

    L.geoJSON(geojson, {
        style: function (feature) {
            return {
                color: "#F012BE",
                weight: 1,
                opacity: 0.1,
                fillopacity: 0.1
            }
        }
    }).bindPopup(function (layer) {
        return `
        <h4> Wald- und Wildschutzzone</h4>
        <h3><a href="${layer.feature.properties.LINK}">Mehr Informationen</a></h3>

        `;

    }).addTo(overlay);
}
loadZones("https://data-tiris.opendata.arcgis.com/datasets/tiris::wald-und-wildschutzzonen.geojson");
```

```
// Implement GPX Track Layer with random Track and Popup
let gpxTrack = new L.GPX(str, {
    async: true,
    marker_options: {
        startIconUrl: "icons/start.png",
        endIconUrl: "icons/finish.png",
        shadowUrl: null,
        iconSize: [32, 37],
        iconAnchor: [16, 37],
    },
    polyline_options: {
        color: "black",
        dashArray: [2, 5],
    },
}).addTo(overlays.gpx);

gpxTrack.on("loaded", function (evt) {
    console.log("Loaded gpx event: ", evt);
    let gpxLayer = evt.target;
    map.fitBounds(gpxLayer.getBounds());

    let popup = `
            <h3>${gpxLayer.get_name()}</h3>
            <ul>
                <li> StreckenLänge: ${(gpxLayer.get_distance()/1000).toFixed()} km</li>
                <li> tiefster Punkt: ${gpxLayer.get_elevation_min()} m</li>
                <li> höchster Punkt: ${gpxLayer.get_elevation_max()} m</li>
                <li> Hoehenmeter bergauf: ${gpxLayer.get_elevation_gain().toFixed()} m</li>
                <li> Hoehenmeter bergab: ${gpxLayer.get_elevation_loss().toFixed()} m</li>`;
    gpxLayer.bindPopup(popup);
}).addTo(map);
```


```
// Wildschutzzonen
async function loadZones(url) {
    let response = await fetch(url);
    let geojson = await response.json();
    // console.log(geojson);
    let overlay = L.featureGroup();
    layerControl.addOverlay(overlay, "Wald- und Wildschutzzonen");
    overlay.addTo(map);

    L.geoJSON(geojson, {
        style: function (feature) {
            return {
                color: "#F012BE",
                weight: 1,
                opacity: 0.1,
                fillopacity: 0.1
            }
        }
    }).bindPopup(function (layer) {
        return
        <h4> Wald- und Wildschutzzone</h4>
        <h3><a href="${layer.feature.properties.LINK}">Mehr Informationen</a></h3>

        `;

    }).addTo(overlay);
}
loadZones("https://data-tiris.opendata.arcgis.com/datasets/tiris::wald-und-wildschutzzonen.geojson");
```

4.3. Implementation der Leaflet-Plugins

Folgende Leaflet-Plugins wurden verwendet:

Leaflet fullscreen plugin,
Leaflet GPX plugin
Leaflet Markercluster
Leaflet MiniMap
Leaflet elevation (optional)

Synchron zur Sommermap mit entsprechenden Winterdaten!

4.4. Implementation des Zufallsgenerators

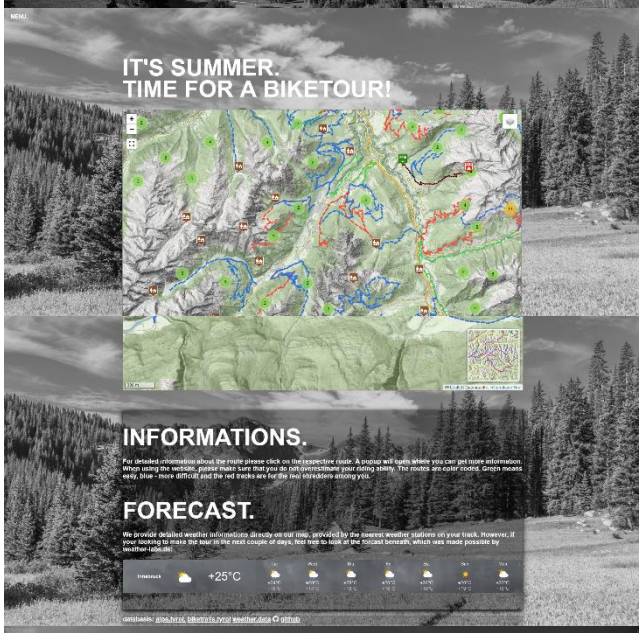
Synchron zur Implementation in der Sommerkarte mit einer Ausnahme. Da die Skitourendaten in einer verschachtelten Ordnerstruktur vorlagen, mussten diese nicht nur nummeriert werden, sondern auch in einen einzigen Ordner gespeichert werden. Da hier nicht nur GPX Dateien vorlagen mussten zudem die GPX Dateien herausgefiltert werden. Hierzu wurde erneut ein Python Skript erstellt.

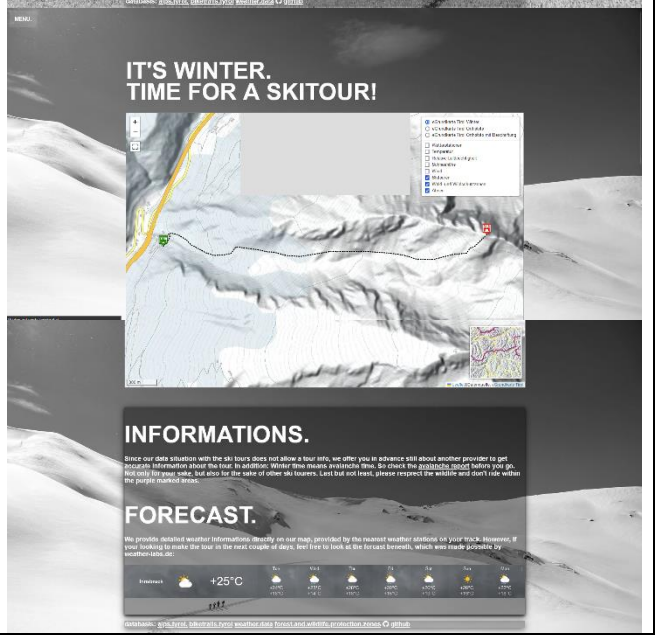
```javascript
// Generate Random Number
let randomNumber = Math.floor(Math.random() * 393);
let strRandom = randomNumber.toString();

// Generate path with RandomNumber
const path = "./data/Skirouten/"
const endPath = ".gpx"
const str1 = path.concat(strRandom)
let str = str1 + endPath
```

```python
import os
import shutil as s

save_path = r'C:\Uni\Webmapping\skitouren.github.io\data\Skitouren\Routen\GPX_Only'

for dictionary in os.listdir("Routen"):
    print(dictionary)
    path = os.path.join(r'c:\Uni\Webmapping\skitouren.github.io\data\Skitouren\Routen', dictionary)
    print(path)
    for file in os.listdir(path):
        if file.endswith('.gpx'):
            completePathSrc = os.path.join(path, file)
            completePathDst = os.path.join(save_path, file)

            s.copy2(completePathSrc, completePathDst)
```

5. Funktionalitätscheck

Der Funktionalitätscheck erfolgt durch einfaches Durchprobieren, ob alle Links Funktionieren und die gewünschte Funktionalität erreicht werden konnte. Dies ist bei allen Pages der Fall. Wir konnten keine Probleme feststellen.





5. Verbesserungsvorschläge

Leider haben bei unserem Projekt nicht alle Funktionalitäten so funktioniert, wie wir uns das vorgestellt haben. Vor allem die Implementation der GeoJson Daten direkt ins HTML und der gewollte Wettercheck zur Routenempfehlung bereiteten uns viel Kopfzerbrechen und mussten schlussendlich aufgegeben werden. Vor diesem Hintergrund würden wir in kommenden Projekten vor allem die Vorarbeit zum Projekt ausweiten. Es hätte deutlich mehr Zeit in die Recherche der Möglichkeiten gesteckt werden sollen, anstatt viel Zeit für unnötigen(-funktionalen) Code zu verschwenden. Außerdem sind wir uns nicht sicher, ob die Karten mit all dem Material überladen sind. Diese Einschätzung liegt schlussendlich aber im Auge des Betrachters.