ERSTELLUNG EINES WEBBASIERTEN, INTERAKTIVEN TAXIWAY-INSTRUCTION GENERATORS

vorgelegt von

Valentin ADLGASSER

Projektarbeit
zur Erlangung der Berufsreifeprüfung
im Fachbereich

INFORMATIONSMANAGEMENT UND MEDIENTECHNIK

BFI-Salzburg

Salzburg, am 13. Mai 2025

vorgelegt bei

Mag. Wolfgang Lehner

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Die vorliegende Arbeit hat in dieser oder einer ähnlichen Form noch keiner anderen Prüfungsbehörde vorgelegen.

.....

Salzburg, 13. Mai 2025 *Unterschrift*

Valentin ADLGASSER

Kurzbeschreibung

Im Rahmen dieser Projektarbeit wurde eine interaktive Webanwendung konzipiert und implementiert, die das Generieren standardisierter Taxi-Instruktionen für sämtliche sechs zivilen Flughäfen Österreichs ermöglicht. Zielsetzung war die Entwicklung eines benutzerfreundlichen digitalen Werkzeugs, das sowohl zur Schulung als auch zur didaktischen Veranschaulichung von Bodenabläufen im Luftfahrtkontext eingesetzt werden kann.

Die technische Umsetzung erfolgte unter Verwendung von HTML, CSS und JavaScript. Für die kartografische Darstellung der Flughafenlayouts und die Integration interaktiver Marker wurde die JavaScript-Bibliothek Leaflet.js herangezogen. Die Anwendung ist responsiv gestaltet und somit auf stationären wie auch mobilen Endgeräten nutzbar.

Nutzerinnen und Nutzer können über eine interaktive Karte einzelne Bereiche des Flughafens – etwa Abstellpositionen, Rollwege, Start- und Landebahnen sowie Holdingpoints – auswählen. Auf Basis dieser Auswahl generiert das System automatisch eine strukturierte Taxi-Instruktion in Textform. Dieses Feature eignet sich insbesondere für die Ausbildung im Bereich Flugfunk und Flugplatzkontrolle.

Die Projektarbeit verbindet theoretische Grundlagen der Webentwicklung mit deren praktischer Umsetzung. Im Verlauf der Realisierung wurden Kenntnisse in der Arbeit mit GeoJSON-Daten, in der Gestaltung interaktiver Kartenlösungen sowie im responsiven Design maßgeblich erweitert. Die modulare Architektur der Anwendung erlaubt zudem eine flexible Erweiterung um zusätzliche Flughäfen und Funktionalitäten.

Inhaltsverzeichnis

1 Einleitung

Im Rahmen dieser Projektarbeit wurde ein praxisnahes Thema aus meinem beruflichen Umfeld gewählt. Die Tätigkeit als Apron Controller am Flughafen Salzburg erfordert täglich die Koordination sicherer und effizienter Bodenbewegungen von Luftfahrzeugen. Dabei stellen sogenannte Taxi-Instruktionen einen zentralen Bestandteil des operativen Betriebs dar.

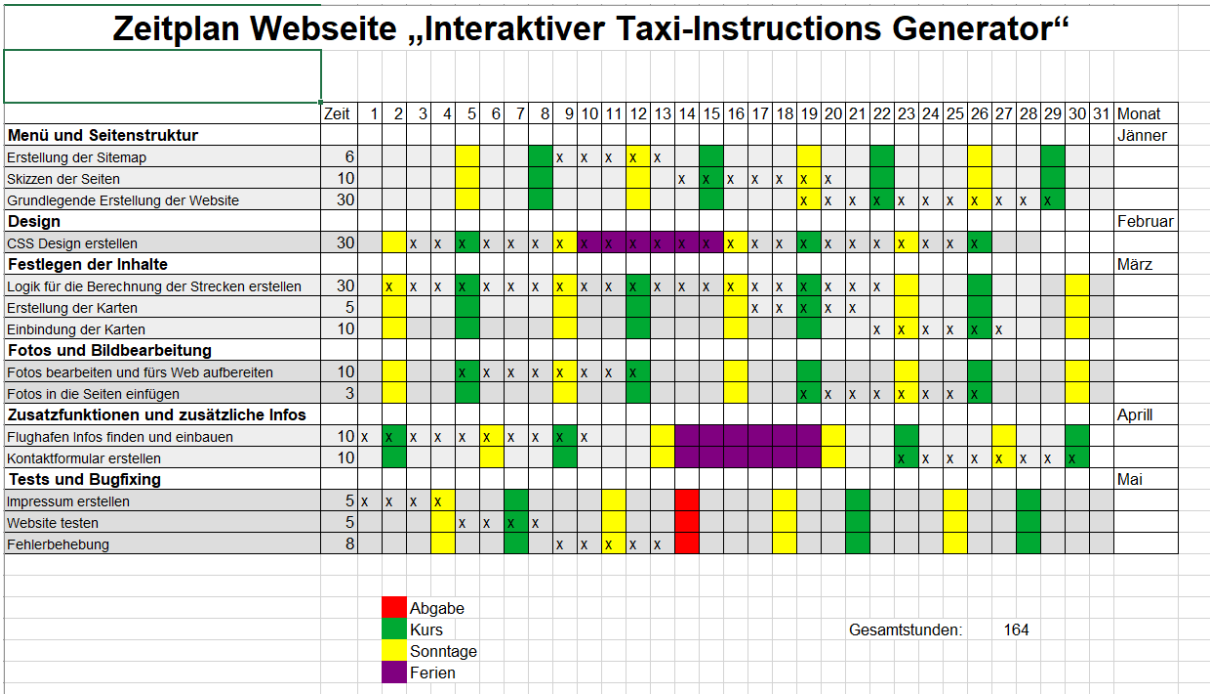
Die Idee zur Entwicklung eines webbasierten Systems entstand aus der täglichen Anwendungspraxis sowie aus der Erfahrung im Umgang mit dem dafür erforderlichen Flugfunkzeugnis. Ziel war es, ein digitales Hilfsmittel zu schaffen, das diesen Arbeitsbereich technologisch unterstützt und dabei sowohl didaktischen als auch operativen Nutzen bietet.

Die entwickelte Anwendung stellt eine interaktive Kartendarstellung zur Verfügung, die die Struktur eines Flughafens visuell abbildet und gleichzeitig automatisch generierte Taxi-Instruktionen in Textform bereitstellt. Diese Kombination aus grafischer Benutzeroberfläche und automatisierter Textausgabe soll das Verständnis für die Abläufe auf dem Vorfeld und den Rollwegen fördern und insbesondere Neueinsteigerinnen und Neueinsteigern im Bereich der Luftfahrt als Lernhilfe dienen.

Die Projektumsetzung ermöglichte eine praxisorientierte Erweiterung der Kompetenzen im Bereich der Webentwicklung. Die Anwendung verbindet dabei fachliche Erfahrung aus der Luftfahrt mit modernen Webtechnologien und demonstriert exemplarisch, wie digitale Tools zur Unterstützung betrieblicher Abläufe konzipiert und umgesetzt werden können.

2 Die Planungsphase

Ein strukturierter Projektablauf ist essenziell für die erfolgreiche Umsetzung komplexer Softwarevorhaben. Daher wurde zu Beginn ein detaillierter Zeitplan erstellt, der sämtliche Arbeitsschritte systematisch gliederte und realistische Zeitfenster für Planung, Umsetzung und Dokumentation definierte. Die zeitliche Planung erfolgte mithilfe einer Excel-Tabelle und wurde in klar abgegrenzte Projektphasen unterteilt.



Zeitplan Webseite „Interaktiver Taxi-Instructions Generator“

	Zeit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	Monat
Menü und Seitenstruktur																																	Jänner
Erstellung der Sitemap	6																																
Skizzen der Seiten	10																																
Grundlegende Erstellung der Website	30																																
Design																																	Februar
CSS Design erstellen	30																																
Festlegen der Inhalte																																	März
Logik für die Berechnung der Strecken erstellen	30																																
Erstellung der Karten	5																																
Einbindung der Karten	10																																
Fotos und Bildbearbeitung																																	
Fotos bearbeiten und fürs Web aufbereiten	10																																
Fotos in die Seiten einfügen	3																																
Zusatzfunktionen und zusätzliche Infos																																	April
Flughafen Infos finden und einbauen	10	x																															
Kontaktformular erstellen	10																																
Tests und Bugfixing																																	Mai
Impressum erstellen	5	x	x	x	x																												
Website testen	5					x	x	x	x																								
Fehlerbehebung	8									x	x	x	x	x																			

Abgabe
Kurs
Sonntage
Ferien

Gesamtstunden: 164

Abbildung 1: Zeitplan

2.1 Zieldefinition

Ziel dieses Projekts war die Entwicklung einer Webanwendung, die auf Basis visuell dargestellter Kartendaten standardisierte Taxi-Instruktionen für sämtliche sechs zivilen Flughäfen Österreichs generiert. Die Anwendung sollte intuitiv bedienbar sein, eine klare Nutzerführung bieten und auf unterschiedlichen Endgeräten – sowohl Desktop- als auch Mobilgeräten – problemlos nutzbar sein.

Zentral war die Anforderung, dass die Anwendung modular aufgebaut ist, sodass eine spätere Erweiterung um zusätzliche Flughäfen ohne strukturelle Änderungen möglich bleibt. Darüber hinaus sollten zu jedem Flughafen grundlegende Informationen in strukturierter Form verfügbar gemacht werden, um das System nicht nur als Werkzeug, sondern auch als Lernhilfe zu positionieren.

2.2 Grobes Designkonzept

Zur Konzeption der Benutzeroberfläche wurde das webbasierte Tool Figma eingesetzt, das sich durch seine intuitive Bedienung und kollaborativen Funktionen auszeichnet. Der Einsatz dieses Werkzeugs ermöglichte es, erste Designentwürfe zu erstellen, zu evaluieren und iterativ weiterzuentwickeln – ohne zusätzliche lokale Softwareinstallation.

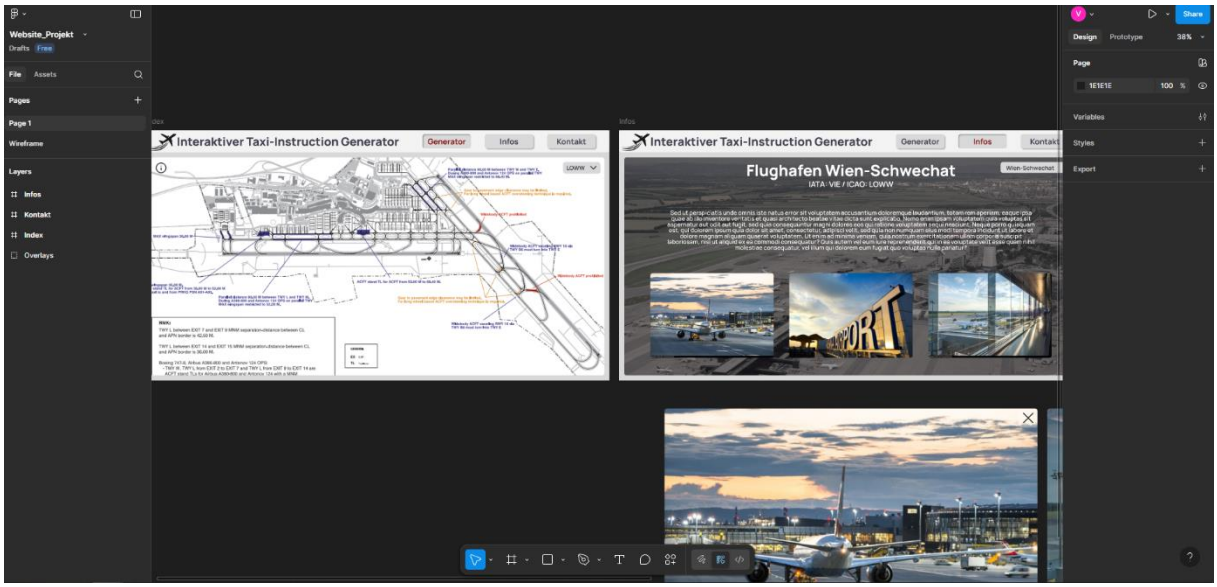


Abbildung 2: Figma

Im Zuge dieser Phase wurden drei Hauptseiten der Anwendung definiert:

2.2.1 Startseite / Generatorseite

Diese Seite stellt die zentrale Funktionalität der Anwendung bereit – die interaktive Kartendarstellung mit Auswahl- und Generierungsfunktion für Taxi-Instruktionen.

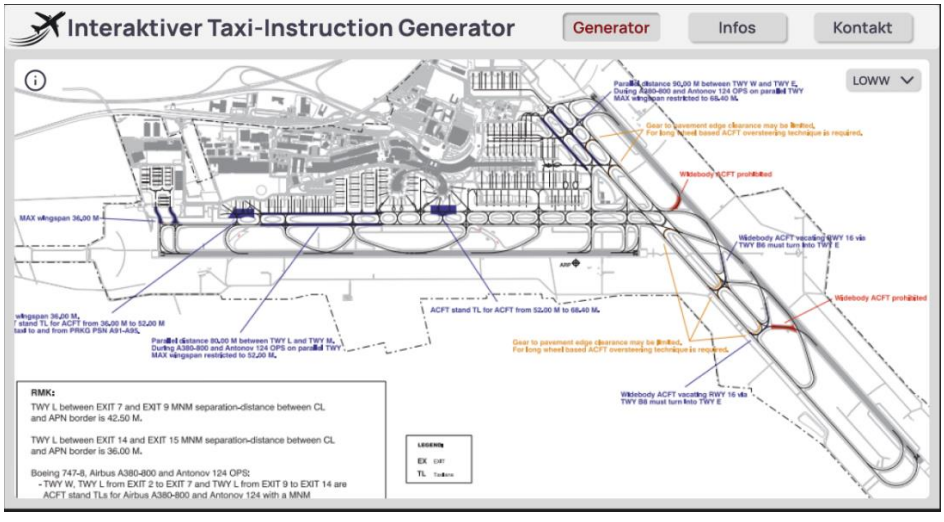


Abbildung 3: Startseite

2.2.2 Informationsseite

Hier werden für jeden der sechs Flughäfen spezifische Inhalte wie Beschreibungstexte und Fotografien bereitgestellt.

Abbildung 4: Infoseite

2.2.3 Kontaktseite

Diese ermöglicht es Nutzerinnen und Nutzern, direktes Feedback zur Anwendung zu geben oder Kontakt mit dem Entwickler aufzunehmen.

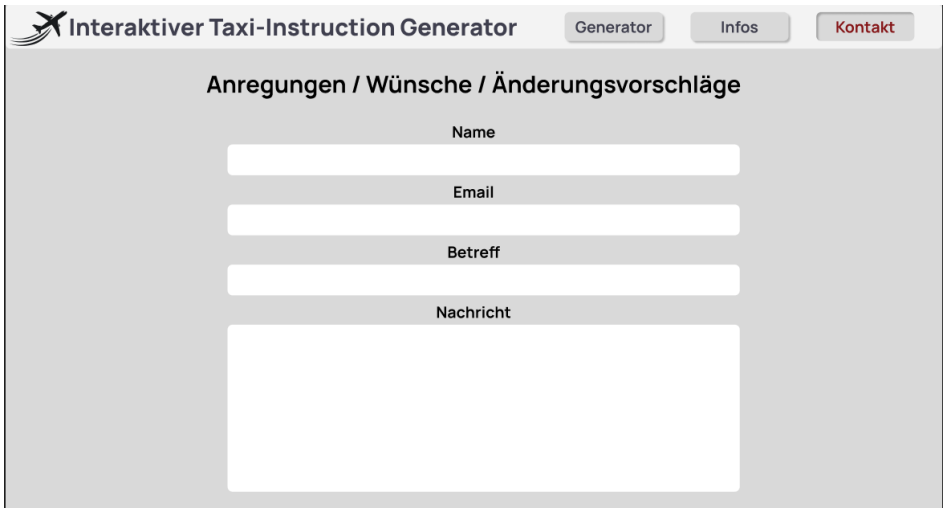


Abbildung 5: Kontaktseite

Die visuelle Gestaltung orientierte sich von Beginn an einem klaren, funktionalen Design mit Fokus auf Übersichtlichkeit und Bedienfreundlichkeit. Während des Projektverlaufs wurden die ursprünglichen Entwürfe regelmäßig angepasst – insbesondere auf Grundlage technischer Machbarkeit und zur Verbesserung der Usability. Die Startseite der Anwendung diente dabei zugleich als zentrale Einstiegsmöglichkeit, während das Layout der Indexseite auf dem der Kontaktseite basierte und daher nicht separat konzipiert wurde.

3 Technische Umsetzung

Nach Abschluss der konzeptionellen Phase wurde mit der praktischen Implementierung der Webanwendung begonnen. Die technische Realisierung erfolgte unter Verwendung etablierter Webtechnologien: HTML zur strukturellen Gliederung, CSS zur Gestaltung der Benutzeroberfläche sowie JavaScript zur Umsetzung interaktiver Funktionen. Ergänzend kam die JavaScript-Bibliothek *Leaflet.js* zur Darstellung der Flughafenlayouts zum Einsatz. Flughafenspezifische Daten wurden in modularen JavaScript-Dateien strukturiert eingebunden.

3.1 Entwicklungsumgebung: Visual Studio Code

Für die Entwicklung des Projekts wurde die quelloffene und plattformübergreifende Entwicklungsumgebung Visual Studio Code (VS Code) verwendet. Diese bietet neben Syntaxhervorhebung und automatischer Codevervollständigung (IntelliSense) auch integrierte Git-Unterstützung sowie eine Vielzahl frei verfügbarer Erweiterungen.

> *„Visual Studio Code ist ein einfacher, aber leistungsstarker Quellcode-Editor, der auf Ihrem Desktop ausgeführt wird und für Windows, macOS und Linux verfügbar ist. Die Lösung bietet integrierte Unterstützung für JavaScript, TypeScript und Node.js und verfügt über ein umfangreiches Ökosystem von Erweiterungen für andere Sprachen und Runtimes (z. B. C++, C#, Java, Python, PHP, Go, .NET).“*
>
> *(„Visualstudio“, 2025)*

Die Entscheidung für VS Code basierte auf dessen Benutzerfreundlichkeit, umfangreichen Anpassungsmöglichkeiten sowie auf bestehenden Vorkenntnissen im Umgang mit diesem Tool.

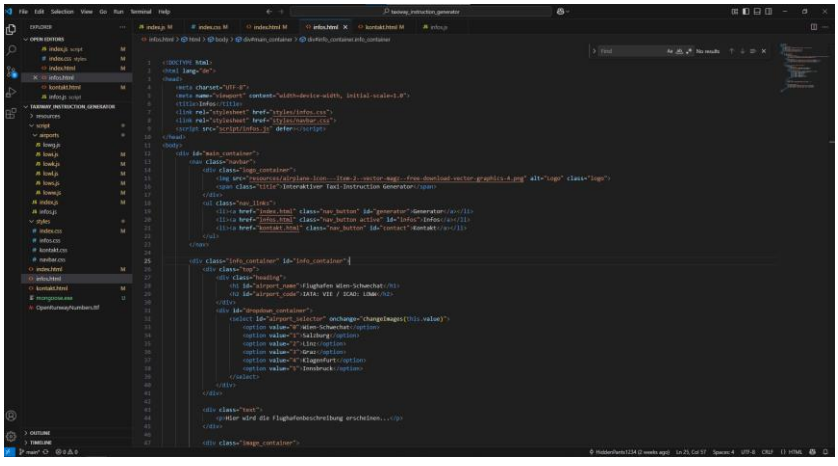


Abbildung 6: Visual Studio

3.2 Strukturierung mit HTML

Die Hypertext Markup Language (HTML) bildet das Fundament jeder Webseite und definiert die semantische Struktur der Inhalte. Im Rahmen dieses Projekts wurden alle sichtbaren Elemente wie Überschriften, Textabschnitte, Formulareingaben, Buttons und Container in HTML umgesetzt.

> *Die Hypertext Markup Language (HTML, englisch für Hypertext-Auszeichnungssprache) ist eine textbasierte Auszeichnungssprache zur Strukturierung elektronischer Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten.*
> *(„Wiki/Hypertext Markup Language“, 2025)*

Besonderes Augenmerk wurde auf eine semantisch korrekte und logisch strukturierte Umsetzung gelegt, um die Lesbarkeit des Quellcodes zu verbessern und die Wartbarkeit der Anwendung langfristig sicherzustellen. Die Inhalte wurden klar vom Design (CSS) und Verhalten (JavaScript) getrennt, wodurch eine modulare Projektarchitektur entstand.

Beispielhaft sieht ein Teil der Startseite folgendermaßen aus:

```
<div class="dropdown-container">
    <label for="airportSelector">Flughafen auswählen:</label>
    <select id="airportSelector">
        <option value="loww">LOWW - Wien</option>
        <option value="lowl">LOWL - Linz</option>
        <option value="lowk">LOWK - Klagenfurt</option>
        <option value="lowi">LOWI - Innsbruck</option>
        <option value="lows">LOWS - Salzburg</option>
        <option value="lowg">LOWG - Graz</option>
    </select>
</div>
```

Abbildung 7: Dropdown

Diese Dropdown-Auswahl wird später über JavaScript mit Funktionalität verbunden.

3.3 Interaktivität mit JavaScript

Zur Umsetzung der Interaktivität wurde die Programmiersprache JavaScript verwendet. Sie steuert sämtliche dynamischen Funktionen der Anwendung – insbesondere die Logik auf der Startseite, welche die Auswahl von Kartenelementen ermöglicht und daraus eine standardisierte Taxi-Instruktion generiert.

> *JavaScript ist eine Programmiersprache, die Entwickler verwenden, um interaktive Webseiten zu erstellen. Von der Aktualisierung von Social Media Feeds bis hin zur Anzeige von Animationen und interaktiven Karten können JavaScript-Funktionen die Benutzerfreundlichkeit einer Website verbessern.*
>
> *(„What is/Javascript“, 2025)*

JavaScript wurde in modularer Form verwendet: Eine zentrale Logikdatei steuert die Benutzerinteraktion, während separate JavaScript-Dateien für jeden Flughafen spezifische Daten (z. B. Markerpositionen, Koordinatenbereiche und Bilddateien) enthalten. Die Trennung von Logik und Daten erhöht die Wiederverwendbarkeit und erleichtert die Erweiterung um zusätzliche Flughäfen.

Ein Auszug aus der Logik zur Marker Interaktion:

```
.on('click', () => {
    const expectedGroup = activeSteps[currentStepIndex];
    if (group !== expectedGroup) {
        alert(`Bitte in der richtigen Reihenfolge wählen: ${expectedGroup.toUpperCase()} ist aktuell dran.`);
        return;
    }

    selected[group] = { name, marker };
    highlightMarker(marker, getHighlightColorForGroup(group));
    currentStepIndex++;

    if (currentStepIndex < activeSteps.length) {
        enableGroup(activeSteps[currentStepIndex]);
    } else {
        showPopup(generateInstructions());
    }
    updateSelectionDisplay();
});
```

Abbildung 8: JavaScript

Durch diese dynamische Steuerung entsteht ein geführter Ablauf, bei dem Nutzende Schritt für Schritt Rollwegpositionen auswählen und am Ende eine automatisch generierte Anweisung erhalten, z.B.:
„Taxi from Main Apron via Exit 12 and Holdingpoint A6 to Runway 29“

3.4 Gestaltung mit CSS

Zur visuellen Gestaltung der Anwendung wurde Cascading Style Sheets (CSS) eingesetzt. Ziel war ein modernes, minimalistisches Erscheinungsbild, das sowohl auf Desktop- als auch Mobilgeräten eine hohe Benutzerfreundlichkeit gewährleistet.

Cascading Style Sheets (CSS) ist eine Programmiersprache, die es Ihnen ermöglicht, das Design von elektronischen Dokumenten zu bestimmen. Anhand einfacher Anweisungen – dargestellt in übersichtlichen Quellcodes – lassen sich so Webseiten-Elemente wie Layout, Farbe und Typografie nach Belieben anpassen.

(„Websites/Webdesign/Was ist CSS“, 2025)

Techniken wie Flexbox und Media Queries wurden verwendet, um ein responsives Design zu ermöglichen. Die Gestaltung der interaktiven Kartenelemente, der Navigationsleiste sowie von Buttons und Formularen erfolgte über ein konsistentes Designsystem. Dadurch wurde ein einheitliches visuelles Erscheinungsbild erzielt, das sich flexibel an verschiedene Bildschirmgrößen anpasst.

Ein Beispiel aus dem Styling der interaktiven Kartenelemente:

```css
#map {
    height: 100%;
    width: 70%;
    border-radius: 10px;
}
```

Abbildung 9: CSS

Diese Anweisung sorgt dafür, dass die Karte auf Desktopgeräten großflächig angezeigt wird, mit abgerundeten Ecken für ein modernes Design. Auch die Navigationsleiste, Buttons und Formulare wurden optisch mit CSS angepasst.

4 Aufbau der Seiten

Die Webanwendung besteht aus mehreren eigenständigen HTML-Seiten, die funktional durch eine einheitliche Navigationsleiste verbunden sind. Jede dieser Seiten erfüllt eine spezifische Aufgabe im Gesamtkonzept der Anwendung. Die klare Trennung zwischen Inhalt, Layout und Funktionalität bildet dabei die Grundlage für eine wartbare und erweiterbare Systemarchitektur.

4.1 Navigationsleiste

Die Navigationsleiste ist auf allen Seiten der Anwendung am oberen Rand positioniert und dient der Benutzerführung. Sie ermöglicht den direkten Wechsel zwischen den Hauptbereichen „Generator“, „Informationen“ und „Kontakt“. Technisch wurde die Navigationsleiste als *<nav>*-Element realisiert und durch eine zentrale CSS-Datei optisch gestaltet.

```
<nav class="navbar">
    <div class="logo_container">
        <img src="resources/airplane_logo.png" alt="Logo" class="logo">
        <span class="title">Interaktiver Taxi-Instruction Generator</span>
    </div>
    <ul class="nav_links">
        <li><a href="index.html" class="nav_button" id="generator">Generator</a></li>
        <li><a href="infos.html" class="nav_button active" id="infos">Infos</a></li>
        <li><a href="kontakt.html" class="nav_button" id="contact">Kontakt</a></li>
    </ul>
</nav>
```

Abbildung 10: Navbar

Das Design umfasst unter anderem die Definition von Farben, Schriftgrößen und Abständen sowie sogenannte Hover- und Active-Zustände. Letztere zeigen an, welche Seite aktuell geöffnet ist. Die Verwendung von Klassen wie *.active* sorgt dabei für konsistente visuelle Rückmeldungen und verbessert die Orientierung innerhalb der Anwendung.

```
.nav_button:hover{
    background-color: #bdbdbd;
}

.nav_button.active{
    background-color: #D9D9D9;
    color: #851416;
    box-shadow: inset 4px 4px 4px 0px rgba(0, 0, 0, 0.25);
}
```

Abbildung 11: Navbar CSS

4.2 Startseite (Generatorseite)

Die Startseite stellt das funktionale Kernstück der Anwendung dar. Sie beinhaltet die interaktive Kartendarstellung sowie die Steuerung der Auswahl- und Generierungslogik für Taxi-Instruktionen. Die Benutzeroberfläche ist in mehrere Bereiche gegliedert:

- Dropdown-Menü zur Auswahl des Flughafens
- Auswahlbereiche für Positions-, Exit-, Holding- und Runwaypunkte
- Kartendarstellung mit interaktiven Markern
- Textfeld mit der generierten Taxi-Instruktion

```
<div id="main_container">
    <div id="map_container">
        <div id="content_wrapper">

            <div id="side_content">
                <div class="dropdown-container">
                    <label for="airportSelector">Flughafen auswählen:</label>
                    <select id="airportSelector">
                        <option value="loww">LOWW - Wien</option>
                        <option value="lowl">LOWL - Linz</option>
                        <option value="lowk">LOWK - Klagenfurt</option>
                        <option value="lowi">LOWI - Innsbruck</option>
                        <option value="lows">LOWS - Salzburg</option>
                        <option value="lowg">LOWG - Graz</option>
                    </select>
                </div>

                <div id="selectionDisplay"></div>

                <button onclick="undoLastStep()">🔙 Letzten Schritt rückgängig machen</button>
            </div>

            <div id="map"></div>

        </div>
    </div>
</div>
```

Abbildung 12: Startseite HTML

Nach Auswahl eines Flughafens wird das entsprechende Kartensegment geladen. Die Flughafendaten sind in separaten JavaScript-Dateien gespeichert, welche die relevanten Koordinaten, Markerinformationen und Bildpfade enthalten. Dies ermöglicht eine klare Trennung zwischen Logik und Daten sowie eine einfache Erweiterbarkeit.

```
class Loww {
    constructor() {
        this.image = './resources/wien_images/LOWW.png';
        this.imageBounds = [[0, 0], [2000, 3000]];

        this.positions = {
            "GAC Apron": [1450, 90],
            "Main Apron": [1320, 750],
            "Maintenance Apron": [1350, 450],
            "Pier West": [1340, 980],
            "Pier East": [1340, 1150],
            "Pier North 1": [1380, 1400],
            "Pier North 2": [1550, 1450],
            "Overflow Apron": [1760, 1350]
        };
```

Abbildung 13: LOWW JS

Die Marker sind interaktiv und erlauben eine schrittweise Auswahl. Jeder Klick wird registriert und entsprechend der vorgesehenen Reihenfolge in die Taxi-Instruktion übernommen.

```
.on('click', () => {
    const expectedGroup = activeSteps[currentStepIndex];
    if (group !== expectedGroup) {
        alert(`Bitte in der richtigen Reihenfolge wählen: ${expectedGroup.toUpperCase()} ist aktuell dran.`);
        return;
    }

    selected[group] = { name, marker };
    highlightMarker(marker, getHighlightColorForGroup(group));
    currentStepIndex++;

    if (currentStepIndex < activeSteps.length) {
        enableGroup(activeSteps[currentStepIndex]);
    } else {
        showPopup(generateInstructions());
    }
    updateSelectionDisplay();
});
```

Abbildung 14: Startseite JS

Die grafische Darstellung erfolgt mit Leaflet.js, wobei statt georeferenzierter Karten statische Bilder mit eigenen Koordinatensystemen verwendet werden. Für jedes Bild wurden Bildgrenzen, Zoom-Stufen und Markerpositionen manuell definiert, um ein exaktes Layout zu gewährleisten.

4.3 Informationsseite

Die Informationsseite bietet vertiefende Inhalte zu den einzelnen Flughäfen. Nutzerinnen und Nutzer können über ein Dropdown-Menü einen Flughafen auswählen, woraufhin dynamisch ein Titel, ein beschreibender Text sowie eine Auswahl von Bildern geladen werden. Die Inhalte sind in Text- und Bilddateien organisiert, die per JavaScript bei Auswahl des Flughafens eingebunden werden.

```html
<div class="image_container">
    <img class="image_box" id="airport_img_1" src="" alt="Image 1">
    <img class="image_box" id="airport_img_2" src="" alt="Image 2">
    <img class="image_box" id="airport_img_3" src="" alt="Image 3">
</div>
```

Abbildung 15: Infoseite HTML

Die Hintergrundfarbe sowie weitere Designelemente passen sich ebenfalls dynamisch an den gewählten Flughafen an, um ein visuell konsistentes Erscheinungsbild zu erzielen. Die CSS-Gestaltung sorgt für ein gleichmäßiges Layout, insbesondere bei der Darstellung der Bildergalerie.

```javascript
document.getElementById("airport_img_1").src = `${base}/image_1.jpg`;
document.getElementById("airport_img_2").src = `${base}/image_2.jpg`;
document.getElementById("airport_img_3").src = `${base}/image_3.jpg`;

document.getElementById("info_container").style.backgroundImage = `url(${base}/background.jpg)`;

document.getElementById("airport_name").textContent = airport.name;
document.getElementById("airport_code").textContent = airport.code;
```

Abbildung 16: Infoseite JS

4.4 Kontaktseite

Die Kontaktseite beinhaltet ein Formular zur direkten Rückmeldung oder Kontaktaufnahme. Die Umsetzung erfolgte über klassische HTML-Formelemente wie *<input>, <textarea>* und *<button>.* Das zugehörige Styling wurde mithilfe von CSS an das Gesamtdesign angepasst.

```
<div class="form">
    <form method="POST" id="email_form">
        <input type="hidden" name="access_key" value="2c70dc37-a3be-42de-94e3-4ab31d705497">
        <div class="row">
            <div class="label">
                <label for="name">Name</label>
            </div>
            <div class="input">
                <input type="text" id="name" name="name" placeholder="Name">
            </div>
        </div>
        <div class="row">
            <div class="label">
                <label for="email">Email</label>
            </div>
            <div class="input">
                <input type="email" id="email" name="email" placeholder="Email">
            </div>
        </div>
```

Abbildung 17: Kontaktseite HTML

Die Formularverarbeitung erfolgt über JavaScript unter Verwendung der Fetch API. Die Daten werden im JSON-Format an den externen Dienst Web3Forms übermittelt, der den E-Mail-Versand übernimmt. Dadurch konnte auf die Einrichtung eines eigenen Mailservers verzichtet werden.

Die JavaScript-Logik zur Formularverarbeitung ist direkt in die HTML-Datei eingebettet, da sie ausschließlich für diese Seite relevant ist und keine Wiederverwendung an anderen Stellen notwendig war. Diese Lösung bietet eine übersichtliche und effiziente Umsetzung für eine klar abgegrenzte Funktion.

Ein Auszug aus dem verwendeten JavaScript:

```
fetch('https://api.web3forms.com/submit', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'Accept': 'application/json'
        },
        body: json
    })
```

Abbildung 18: Kontaktseite JS

4.5 Impressum

Die Impressum-Seite enthält alle rechtlich erforderlichen Angaben zum Betreiber der Webanwendung. Sie ist ausschließlich über die Kontaktseite erreichbar und wird aus dem Hauptmenü bewusst ausgeklammert, um den Fokus der Navigation auf funktionale Inhalte zu lenken.

Die Seite informiert in strukturierter Form über:

- Anbieterinformationen (Name, Adresse, Kontakt)
- Regulierungsbehörde (KommAustria)
- Hinweise zur EU-Streitschlichtung
- Haftungsausschlüsse für Inhalte und externe Links
- Urheberrechtsbestimmungen

Das Layout ist minimalistisch und zentriert, orientiert sich jedoch visuell am Gesamtstil der Anwendung. Die Navigationsleiste sowie das responsive Menüverhalten sind analog zu den übrigen Seiten implementiert.

Durch die Bereitstellung dieser Seite erfüllt die Webanwendung die Vorgaben der gesetzlichen Anbieterkennzeichnung gemäß E-Commerce-Gesetz (ECG).

```
<body>
    <nav class="navbar">
        <div class="logo_container">
            <img src="resources/airplane_logo.png" alt="Logo" class="logo">
            <span class="title">Interaktiver Taxi-Instruction Generator</span>
        </div>
        <div class="hamburger" onclick="toggleMenu()">☰</div>
        <ul class="nav_links" id="navLinks">
            <li>
                <a href="index.html" class="nav_button" id="generator">Generator</a>
            </li>
            <li>
                <a href="infos.html" class="nav_button" id="infos">Infos</a>
            </li>
            <li>
                <a href="kontakt.html" class="nav_button" id="contact">Kontakt</a>
            </li>
        </ul>
    </nav>
    <div class="impressum">
        <h1>IMPRESSUM</h1>
        <h3>Informationen über den Diensteanbieter</h3>
        <p>Valentin Adlgasser</p>
        <p>Dorf 32b / 13 <br>
        5301 Eugendorf <br>
```

Abbildung 19: Impressum HTML

5 Sonstige verwendete Technologien und Programme

Im Zuge der Umsetzung dieser Webanwendung kamen zusätzlich zu den Basistechnologien HTML, CSS und JavaScript weitere Werkzeuge und Bibliotheken zum Einsatz, die spezifische Anforderungen abdeckten. Diese unterstützten insbesondere die Visualisierung, Bildbearbeitung sowie das Testen der Anwendung unter realistischen Bedingungen.

5.1 Leaflet.js – Interaktive Kartendarstellung

Die Bibliothek *Leaflet.js* wurde für die Darstellung der interaktiven Karten eingesetzt. Es handelt sich dabei um eine moderne, leichtgewichtige JavaScript-Bibliothek, die insbesondere für mobile und performante Kartenanwendungen konzipiert wurde. Im Unterschied zu klassischen Geoinformationssystemen wurde *Leaflet* hier zur Darstellung von statischen Flughafenplänen verwendet, auf denen Marker dynamisch platziert werden.

Die Marker repräsentieren konkrete Positionen wie Abstellflächen, Exits, Holdingpoints und Start-/Landebahnen. Durch schrittweise Auswahl dieser Punkte wird eine vollständige Taxi-Instruktion generiert. Die Interaktion erfolgt dabei direkt auf dem dargestellten Bild, das in ein individuelles Koordinatensystem eingebettet wurde.

Die Entscheidung für Leaflet fiel aufgrund der hohen Flexibilität, umfangreichen Dokumentation und der Möglichkeit, nicht-georeferenzierte Karten effizient zu integrieren. Darüber hinaus unterstützt Leaflet eine modulare Erweiterung durch Plugins, was zukünftige Funktionsausbauten erleichtert.

```
function addMarkers(locations, color, group) {
    for (const [name, coords] of Object.entries(locations)) {
        const marker = L.circleMarker(coords, {
            radius: 10,
            color,
            fillColor: color,
            fillOpacity: 0.5,
        })
        .addTo(map)
        .bindTooltip(name, { permanent: false, direction: 'top' })
        .on('click', () => {
            const expectedGroup = activeSteps[currentStepIndex];
            if (group !== expectedGroup) {
                alert(`Bitte in der richtigen Reihenfolge wählen: ${expectedGroup.toUpperCase()} ist aktuell dran.`);
                return;
            }

            selected[group] = { name, marker };
            highlightMarker(marker, getHighlightColorForGroup(group));
            currentStepIndex++;

            if (currentStepIndex < activeSteps.length) {
                enableGroup(activeSteps[currentStepIndex]);
            } else {
                showPopup(generateInstructions());
            }
            updateSelectionDisplay();
        });

        marker.options.group = group;
        marker.options.originalColor = color;
    }
}
```

Abbildung 20: Leaflet

5.2 GIMP – Bildbearbeitung

Zur grafischen Aufbereitung der Flughafenpläne sowie zur Bearbeitung der Bilder auf der Informationsseite wurde das Open-Source-Programm GIMP (GNU Image Manipulation Program) verwendet. Es bietet professionelle Werkzeuge zur Bildbearbeitung und gilt als leistungsstarke Alternative zu kommerzieller Software wie Adobe Photoshop.

Im Rahmen des Projekts wurde GIMP unter anderem für folgende Aufgaben verwendet:

- Erstellung und Bearbeitung der grafischen Flughafenübersichten
- Zuschneiden und Freistellen einzelner Kartenelemente
- Anpassung von Helligkeit, Kontrast und Farbbalance zur besseren Lesbarkeit
- Export in geeignete Formate und Auflösungen für die Webdarstellung

Durch den Einsatz dieses Werkzeugs konnte ein einheitlicher und klar strukturierter visueller Stil gewährleistet werden, der sich positiv auf die Benutzerfreundlichkeit der Anwendung auswirkt.

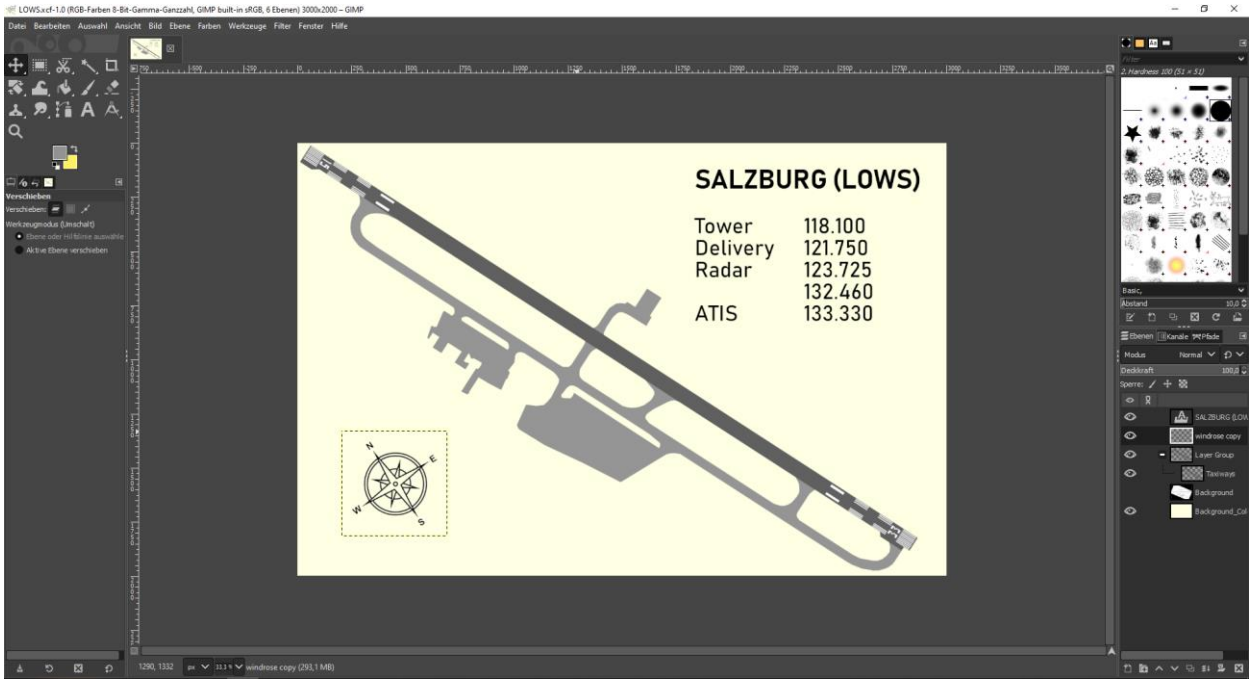


Abbildung 21: GIMP

5.3 Moongoose.exe – Lokaler Webserver

Während der Entwicklungs- und Testphase wurde die Anwendung lokal im Browser getestet. Um dabei typische Browser-Sicherheitsmechanismen wie die *CORS-Richtlinien* (Cross-Origin Resource Sharing) zu umgehen, kam die Software *moongoose.exe* als lokaler Webserver zum Einsatz.

Bei direktem Öffnen von HTML-Dateien über den Dateipfad (*file://*) blockieren moderne Browser aus Sicherheitsgründen den Zugriff auf externe Ressourcen, wie z. B. lokale Bilddateien oder Textdateien. Dies betraf insbesondere die Info-Seite, auf der Inhalte dynamisch per JavaScript nachgeladen werden sollten. Durch den Start eines lokalen Webservers konnte die Anwendung stattdessen über *http://localhost* ausgeführt werden, wodurch die Einschränkungen aufgehoben wurden.

Der Einsatz eines lokalen Servers simuliert realitätsnahe Einsatzbedingungen und stellt sicher, dass sämtliche Funktionen – insbesondere das dynamische Nachladen von Inhalten – korrekt und zuverlässig getestet werden können.

6 Herausforderungen & gewonnene Erkenntnisse

Die Entwicklung einer interaktiven Webanwendung im luftfahrtspezifischen Kontext brachte zahlreiche fachliche, technische und konzeptionelle Herausforderungen mit sich. Diese stellten nicht nur konkrete Aufgaben im Rahmen der Projektarbeit dar, sondern ermöglichten auch einen erheblichen Wissenszuwachs in verschiedenen Bereichen der Webentwicklung.

Eine der ersten Herausforderungen bestand in der Auswahl und Strukturierung der darzustellenden Inhalte. Bereits in der Planungsphase war zu klären, welche Informationen zu den Flughäfen aufgenommen werden sollten, wie viele Auswahloptionen technisch sinnvoll umsetzbar sind und in welchem Umfang Taxi-Instruktionen erzeugt werden können, ohne die Nutzerfreundlichkeit zu beeinträchtigen. Hier zeigte sich die Notwendigkeit einer konsequenten Reduktion auf das Wesentliche, um eine intuitive Bedienung der Anwendung sicherzustellen.

Ein weiteres wesentliches Lernfeld ergab sich beim Einsatz von *Leaflet.js*. Während die Einbindung georeferenzierter Karten gut dokumentiert ist, erforderte die Verwendung statischer Grafiken mit benutzerdefinierten Koordinatensystemen ein erhebliches Maß an individueller Anpassung. Für jeden Flughafen mussten exakte Bildabmessungen, Zoom-Stufen sowie präzise Marker-Koordinaten definiert werden. Diese Aufgaben erforderten sorgfältiges Arbeiten, wiederholtes Testen und ein gutes Verständnis für kartografische Zusammenhänge.

Auch das dynamische Nachladen von Texten und Bildern auf der Informationsseite stellte anfangs eine technische Hürde dar. Aufgrund der *CORS*-Beschränkungen blockierten Browser den Zugriff auf lokale Dateien. Erst durch die Integration eines lokalen Webservers konnte die Funktionsweise realitätsnah simuliert und sichergestellt werden, dass alle Inhalte wie vorgesehen geladen wurden.

Im Bereich der JavaScript-Entwicklung zeigte sich die Bedeutung einer klaren Code-Struktur und durchdachter Logik. Der Auswahlprozess von Rollpositionen, Exits, Holding Points und Runways musste so gestaltet werden, dass fehlerhafte oder widersprüchliche Instruktionen ausgeschlossen sind. Die Entwicklung eines robusten und logischen Auswahlmechanismus vermittelte ein tiefgehendes Verständnis für ereignisgesteuerte Programmierung, Fehlerbehandlung und Benutzerführung.

Gestalterisch stellte sich die Aufgabe, eine moderne und zugleich funktionale Benutzeroberfläche zu schaffen. Ziel war ein aufgeräumtes, ansprechendes Layout, das auch auf mobilen Endgeräten zuverlässig funktioniert. Mithilfe von GIMP konnten visuelle Elemente einheitlich aufbereitet und sinnvoll in das Design integriert werden – sowohl für die Kartendarstellung als auch für die Flughafenbilder.

Insgesamt war das Projekt nicht nur eine technische Umsetzung, sondern auch eine konzeptionelle und gestalterische Auseinandersetzung mit einem fachlich relevanten Thema. Die erworbenen Kenntnisse in HTML, CSS, JavaScript sowie im Umgang mit modernen Entwicklungswerkzeugen führten zu einem deutlich erweiterten Verständnis für die Schnittstellen zwischen Gestaltung, Funktion und technischer Realisierung. Besonders hervorzuheben ist die Erfahrung, ein vollständiges Softwareprojekt eigenverantwortlich von der Planung bis zur Umsetzung durchzuführen – ein zentrales Element praxisorientierten Lernens.

Abbildungsverzeichnis

Literarturverzeichnis

„Visualstudio“. (11.05.2025). Visualstudio. Von https://visualstudio.microsoft.com/de/ abgerufen

„Websites/Webdesign/Was ist CSS”. (12.05.2025). IONOS. Von https://www.ionos.at/digitalguide/websites/webdesign/was-ist-css/ abgerufen

„What Is/Javascript“. (12.05.2025). Amazon. Von https://aws.amazon.com/de/what-is/javascript/ abgerufen

„Wiki/Hypertext Markup Language“. (11.05.2025). Wikipedia. Von https://de.wikipedia.org/wiki/Hypertext_Markup_Language abgerufen

Anlage

Projektantrag