PathArt – GPS Art Route Generator
Web Application Development

Introduction:

This Design Specification outlines the architecture, data flow, modules, and implementation details for the PathArt application. PathArt transforms user-uploaded sketches into GPS routes that follow real-world streets. The system integrates vectorisation, geospatial mapping, routing APIs, and geographic data export.

System Architecture

PathArt uses a three-layer architecture:

Frontend

- Runs in browser
- Technologies: HTML, CSS, JavaScript
- Mapping: Leaflet.js
- Vectorisation: Potrace.js / Paper.js

Handles:

- File upload
- Image vectorisation
- Map interaction
- Shape placement, scaling, rotating
- Sending sampling points to backend

Backend

Backend Framework: Python Flask

Responsible for:

- Receiving sampled coordinates
- Calling routing APIs (ORS / Mapbox)
- Combining partial routes
- Converting GeoJSON to GPX
- Securing API keys

REST Endpoints:

- POST /generate-route
- POST /export-gpx

External Services

- OpenRouteService Directions API – routing engine
- OpenStreetMap – map tiles
- Mapbox Directions API (optional alternative)

System Data Flow:

Step-by-Step Process

1. User uploads sketch
   → Frontend reads raster image.
2. Vectorisation
   → Potrace.js converts image → SVG path.
   → Paper.js optionally simplifies line.
3. User positions vector on map
   → Map overlay displays SVG path.
4. Sampling
   → Frontend extracts coordinate points from SVG relative to map position.
5. Routing requests
   → Frontend sends pairs of points to backend.
   → Backend calls ORS for each segment.
6. Route merging
   → Backend merges GeoJSON segments into one polyline.
7. Display result
   → Frontend displays final route on map.
8. Export
   → User exports GPX file.

Component Design:

Frontend Components

Image Upload Component

- Accepts PNG/JPEG
- Displays preview
- Sends to vectorisation module

Vectorisation Module

- Potrace.js extracts outlines
- Outputs SVG path
- Allows simplification level (low/medium/high)

Map Interface (Leaflet)

- Displays OpenStreetMap raster tiles
- Places SVG overlay
- Supports transformations:
  - Drag
  - Scale
  - Rotate

Sampling Module

- Samples N points along vector
- Converts to lat/lon using map projection

- Sends JSON to backend

Route Display Module

- Renders GeoJSON route
- Shows optional distance/time estimate

Export Module

- Sends route to backend
- Receives GPX
- Triggers download

Backend Components

Route Generator

- Accepts array of coordinate pairs
- For each pair:
  - Sends request to ORS Directions
  - Extracts returned GeoJSON
- Appends all segments
- Ensures continuity between segments

GPX Converter

- Accepts GeoJSON
- Uses gpxpy (Python) to convert
- Returns .gpx file

API Key Manager

- Reads keys from environment variables
- Never exposed to fronten

Routing Algorithm Design:

Input

- Ordered list of sampled coordinate points
- Routing profile (walk/run/cycle)

Algorithm

For i = 0 → N-2:

1. Take point[i] and point[i+1]
2. Send to ORS Directions API
3. Receive route segment as GeoJSON
4. Merge into master route

Output

- Final GeoJSON polyline representing the entire path

Failure Handling

- If a segment fails:
  - Retry with nearby point
  - Notify user of “unroutable section”

Vector Processing Design:

Simplification (Paper.js)

- Removes unnecessary points
- Smooths curves
- Reduces routing load

Ramer–Douglas–Peucker (optional)

- Tolerance-based simplification
- Used if performance needed

User Interface Design:

Layout Structure

Top Section:

- Title + brief instructions
- Upload button
- Shape simplification slider

Middle Section:

- Fullscreen map
- Interactive shape overlay

Side Panel:

- Generate Route
- Routing mode selector
- Distance/time display
- Export GPX button

Technology Stack:

Layer	Technology
Frontend	HTML, CSS, JavaScript, Leaflet.js, Potrace.js
Backend	Python Flask
Routing	OpenRouteService / Mapbox
Data Formats	SVG, GeoJSON, GPX

Constraints

1. Free ORS tier limits routing requests
2. High-detail sketches may create too many sampling points
3. Internet connection required
4. CPU load on client for vectorisation

Future Design Improvements:

1. AI-powered sketch cleaning
2. Automatic suggestion of best map area
3. Multi-user collaborative drawing
4. 3D elevation path rendering
5. Real-time route animation playback