**Project Risks**

**High Risks**

**1.**

**Scope Creep**

* **Expanding Beyond Initial Objectives**: As the project evolves, there’s a risk that additional features or criteria will be requested, such as including new walkability factors or supporting more user types (e.g., tourists vs. residents). This can lead to project delays and resource overrun, as the team will need to reallocate time and budget to accommodate the new requests.
* **Overcomplicating User Interface**: Trying to include every possible walkability factor might lead to a cluttered or overly complex user interface, which would not only increase development time but also make the tool harder to use. This complexity could turn off users and negatively affect user engagement.
* **Difficulty in Quantifying Subjective Criteria**: Certain walkability factors, like “visual beauty” or “spaces for social interaction,” are subjective and challenging to quantify. Developing reliable metrics for such criteria can be difficult, and inconsistent implementation could reduce the tool’s reliability and user trust.

**2.**

**Technical Feasibility**

* **Complex Algorithm Development**: Developing algorithms that integrate various criteria (e.g., population density, setbacks, greenery, etc.) into a cohesive and computationally efficient model can be a major technical challenge. Balancing accuracy with performance is critical, and inefficient algorithms could lead to performance bottlenecks.

**3**.

**Performance Risks**

* **Tool Optimization**: Ensuring the tool runs efficiently, even with heavy data loads like detailed maps or high user activity, can be a challenge. If not optimized, the tool could experience slowdowns or crashes, reducing its usability and appeal to users.

**4**.

**Resource Constraints**

* **Time and Budget Overruns**: The complexity of developing a sophisticated tool with multiple integrated features can lead to underestimations of both time and resources required. This could cause project deadlines to be missed and the project to go over budget.
* **Insufficient Testing Time**: If not enough time is allocated for testing and debugging, the tool could be released with flaws or usability issues, leading to a poor user experience. Insufficient testing may also mean that important feedback is missed, requiring costly post-launch fixes.

**5.**

**Team Skill Gaps**

* **Lack of Expertise in Specialized Areas**: Developing a tool that involves urban planning data, GIS integration, and interactive visualization may require specialized skills. If the team lacks expertise in these areas, it may result in delays or poor implementation of key features, potentially jeopardizing the success of the project.

**Medium Risks:**

**1.**

**User Interaction Risks**

* **Unintuitive User Interface**: If the tool’s interface is not intuitive, users may struggle to input their preferences, leading to frustration and potentially skewed data. Poor UI design can lower user engagement and result in reduced participation or poor-quality feedback.
* **Complexity of User Interaction Scenarios**: Accommodating two distinct scenarios (leisure vs. utilitarian walking) in a simple, user-friendly manner could be challenging. If the tool’s interface doesn’t effectively separate or clarify these scenarios, it could confuse users and result in inaccurate or mixed feedback.

**2.**

**Third-Party Dependencies**

* **Dependency on External Libraries/Tools**: The project may rely on external tools like mapping APIs or data visualization libraries. If these tools have bugs, compatibility issues, or become obsolete, it could delay development or necessitate additional work to find alternatives.

**3.**

**Testing and User Feedback Risks**

* **Insufficient User Testing**: If the tool is not thoroughly tested with actual users, developers may miss critical feedback about usability or accuracy. This could lead to a tool that technically works but fails to meet user expectations or doesn’t engage the target audience effectively.
* **Feedback Incorporation**: If user feedback is gathered late in development, implementing necessary changes could cause significant delays and require major rework. Incorporating feedback early and throughout development is crucial to avoid last-minute disruptions.

**Low Risks:**

**1.**

**Team Skill Gaps**

* **Learning Curve**: Developers may need to learn new tools or frameworks during the project, which can temporarily slow down progress. However, this is often manageable with good planning and should only cause minor delays if training is done in the early stages.