Risk Analysis by Team CyberPunk

1. Identify the different types of risks that can occur in a software engineering project. This includes **technical risks**, such as **software defects** or **hardware failures**, as well as **non-technical risks**, such as **budget overruns or schedule delays**, **knowledge and skill deficits**.
2. Develop a method for assessing the likelihood and impact of each risk. This could involve using qualitative or quantitative techniques, or a combination of both.

---

1. **Unmet Expectations:** This is a risk that occurs when the customer or stakeholder expectations for the project are not met. This can be caused by poor communication, unrealistic expectations (which is very common for startups), or changes to the project scope.
    a. To assess the likelihood and impact:
        i. Assess the **experience level** of the team members
        ii. Consider the **realism** of the customer's expectations
        iii. Consider the level of **communication** between the developers and the customers/stakeholders
        iv. Assess the clearness of the project's **requirements**
    b. To mitigate these limitations, the team can:
        i. **Document the project requirements:** Clearly document the requirements to help ensure that everyone has a shared understanding of what is expected of the project.
        ii. **Gather feedback:** Encourage users to complete surveys and/or rating scales on their satisfaction with the app. This feedback serves to provide valuable information for identifying and refining app features and functionalities in order to continuously meet user expectations
2. **Schedule Delays**
    a. To assess the likelihood and impact:
        i. **Historical data:** How long have projects like this previously taken a team? How reliable and productive have each team member been in the past?
        ii. **Consider the scope:** Carefully discuss the complexity and scope of the project beforehand and for each newly planned feature. Overly ambitious tasks can lead to major setbacks.
    b. To mitigate these limitations, the team can:
        i. **Integrate Project Management Tools:** Tools such as Trello can be used to assign and monitor the progress of development tasks, on top of regular team meetings to evaluate potential setbacks, such as work load and learning curves.
        ii. **Create a detailed schedule** for the project plan that the team can adhere to. When delays occur, try to identify the root cause and take the necessary steps to prevent it from happening again.

3. **Compatibility Issues/Platform Limitations:** When the capabilities of a platform prevent it from supporting a particular application or functionality, it is a platform limitation. For example, making an iOS build on a Windows machine requires a Mac computer and an Apple Developer account.
    a. To assess the likelihood and impact:
        i. Survey the team: Before starting development, figure out what platforms each developer will be using
        ii. Consult experts or experienced developers or project managers to get their assessment on the matter
        iii. Read the documentation of the framework and pay attention to the specific minimum requirements as well as the compatibility limitations that may be mentioned.
    b. To mitigate these limitations, the team can:
        i. Prepare the workstations that will make the builds beforehand
        ii. Share different platform systems among different team members
        iii. Develop using a multiplatform framework
4. **Lack of Ownership:** This is a risk that occurs when no one is clearly responsible for a particular task or aspect of a project. This can lead to confusion, delays, and poor quality work.
    a. To assess the likelihood and impact:
        i. Pay attention to the clarity of the project roles and responsibilities. Are they specifically assigned to people and are the boundaries clearly defined?
        ii. Assess the level of communication and collaboration within the project team
    b. To mitigate these limitations, the team can:
        i. **Team Communication:** Make a point to emphasize the importance of accountability and communication by addressing related issues that could arise, such as poor code quality and incompletion of tasks, and how this could impact the progress of the app and other team member's workload
        ii. Clearly define roles and responsibilities of each team member
        iii. Have accountability for members of the team that fail to deliver on their promises
        iv. Regularly inspect the project to make sure everyone is on track and there's no areas where ownership is unclear. Multiple members doing the same tasks is unproductive. It is better for one person's work to be used for all.

5. **Scalability concerns:**
    a. To assess the likelihood and impact:
        i. **Performance Testing:** test performance by varying loads so that bottlenecks and limitations are identified.
        ii. **Traffic Analysis:** the team could analyze anticipated and current traffic patterns to better understand peak loads and how the user behaves.
        iii. **Profiling and Monitoring:** the team could use profiling tools such as Intel Vtune or VisualVM to analyze CPU performance. They could also use Valgrind for memory leaks and Dtrace to help in tracing system calls and monitor I/O activities.
    b. To mitigate these limitations, the team can:
        i. **Optimizing code and Algorithms:** the team could look over inefficient algorithms to improve efficiency.
        ii. **Regular Review:** review the system architecture to make sure that it aligns with any technological advancements or demands that come up.
        iii. **Parallelism:** languages such as python have multiprocessing libraries that enable parallel processing by using multiple CPU cores to execute tasks simultaneously; this in turn could really speed up data analysis tasks.