Risk Management Plan

Open Source Intelligence (OSINT) Application

Members: Steve Andraws, Matthew Kall, Lucas Jordan

Professor Dr. Maurice Dawson Jr.

- **1 Introduction**

Managing risks in this project wasn't just about preparing for worst case scenarios, it was about building resilience in every part of our process. We knew going in that no project involving live APIs, GUI logic, and multiple collaborators was not going to be perfectly smooth. That's why this Risk Management Plan became such a central part of how we worked.

The purpose of this plan was to identify early warning signs, reduce uncertainty, and give us a framework to respond calmly when something did go off track and it did. The scope of this plan covered all major project phases, from planning and development through testing and final delivery. Our approach was agile but grounded in PMI best practices: we logged risks, rated them, reviewed them weekly, and assigned owners to each one.

- **2 Roles and Responsibilities**

Clear roles were key to avoiding confusion. We didn't just say someone will fix it we gave names to every task, especially when it came to managing risk. Here's who did what:

Name	Role	Responsibility
Steve Andraws	Project Manager	Maintained the risk register, managed Planner tasks, and led reassignment if delays occurred.
Matthew Kall	Lead Developer	Oversaw GUI testing, managed merge approvals, and built fallback tools for critical functions.
Lucas Jordan	API/Data	Monitored API access limits, rotated keys, and maintained backup data sources.

This clarity helped us react fast when risks emerge, no one asked who's handling this? We already knew.

- **3 Risk Management Processes**

We built our risk process like we built our app: modular, flexible, and tested at each stage.

- **3.1 Identify Risks**

Risk spotting wasn't a onetime thing, it was a habit. During planning, we used brainstorming and past project notes to list obvious risks (merger conflicts, API quota errors). As the project evolved, we added more: GUI crashes, timeline slipups, and team fatigue. Everyone contributed, and no suggestion was too small. That openness helped us catch blind spots early.

- **3.2 Risk Register**

Our Risk Register lived in a shared Excel file on OneDrive simple, visible, and easy to update. Each row tracked the risk name, owner, likelihood, severity, mitigation strategy, and status. Color coded formatting helped us quickly spot which items needed immediate attention during meetings.

- **3.3 Analyze Risks**

We scored each risk based on:

- **Likelihood**: How likely it was to happen
- **Impact**: What damage it could have caused if it did

That gave us a Risk Level (Low/Medium/High/Extreme) which helped prioritize focus. High risk items like API key expiration were flagged for weekly check-ins, while medium risks like presentation AV issues were reviewed every few days near the deadline.

- **3.4 Risk Response Planning**

We didn't wait for things to break we planned for what we'd do if they did. Strategies included:

- **Avoidance**: Matt refactored code to prevent fragile integration points.
- **Mitigation**: Lucas rotated API keys proactively instead of waiting for errors.
- **Acceptance**: For rare risks with low impact, we acknowledged the risk but didn't act until needed.
- **Contingency**: Steve prerecorded our final demo in case our live presentation glitched and it nearly did.

We didn't overengineer solutions, but we always had a backup.

- **3.5 Risk Monitoring and Control**

Every week, we reviewed the Risks We checked:

- Has the likelihood changed?
- Is the mitigation working?
- Does someone need help?

This rhythm helped us fix problems before they escalated. It gave us confidence to build fast, knowing we had support when things (inevitably) got messy.