Risk Management Plan: Kirana App

How We Found the Risks

We used a simple and practical approach to identify potential problems for the Kirana App project.

Our method combined three things:

- **Team Experience:** We used our own knowledge of the local Karachi market and technology to brainstorm what could go wrong.
- **Historical Data:** We looked at the common reasons why other local apps in Pakistan have struggled to compete with bigger platforms.
- **Standard Checklists:** We used well-known software project risk checklists (like Boehm's Top 10) to make sure we considered common issues like personnel shortfalls or building the wrong features.

Main Project Risks & Our Plan to Handle Them

This table lists the most important risks and how we plan to manage them.

Risk Description	Avg. Likelihood (1-5)	Avg. Impact (1-5)	How We'll Handle It (Mitigation Strategy)
Personnel Shortfalls: Limited development team size could cause bottlenecks if key members are unavailable.	3.6	3.2	Ensure key knowledge is shared through documentation and weekly meetings. Cross-train where possible.
Unrealistic Schedules: The aggressive 2-month timeline may not be enough for complex tasks like API integration.	3.8	4	Strictly follow the WBS and monitor progress weekly. If delays occur, focus only on essential MVP features.
Developing Wrong Functions: The app may miss critical marketplace or chat	3.2	2.8	Validate features against the SRS and get user feedback early during UAT.

features that users actually expect.			
Wrong User Interface: The UI/UX may not be intuitive for the target audience, leading to poor adoption.	3.4	2.4	Develop a clickable Figma prototype and get feedback from target users before starting development.
Gold Plating: Developers might add unnecessary features beyond the MVP scope, wasting time.	2.6	2	Enforce the Change Control Process. All new features must be justified and formally approved.
External Component Failure: Third-party services or libraries (like Firebase) may have outages or not work as expected.	4.6	1.8	Choose reliable, well-documented services. Have contingency plans for critical dependencies where possible.
Performance Shortfalls: The real-time chat and marketplace updates may lag under load.	3.4	2	Use Firebase's scalable infrastructure. Conduct load testing to identify and fix performance bottlenecks.
Organizational Risk: A team member losing motivation or not communicating can create knowledge gaps.	2.4	2.8	Hold regular weekly meetings to keep everyone engaged and informed. Use shared documentation on GitHub.
Technical Integration Risk: Issues may arise when connecting the Next.js frontend with the Firebase backend and APIs.	4	3	Define clear API contracts early in the process. Conduct integration testing as soon as components are ready.

Team’s Rating

Risk Description	Moin		Arib		Haider		Riyan		Haris	
	Impact	Likely Hood	Impact	Likely Hood	Impact	Likely Hood	Impact	Likely Hood	Impact	Likely Hood
Personnel Shortfalls	3	3	4	3	3	3	4	3	4	4
Unrealistic Schedules	4	4	4	4	4	4	3	4	4	4
Developing Wrong Functions	2	3	3	2	3	3	4	2	4	4
Wrong User Interface	4	3	3	2	3	2	4	2	3	3
Gold Plating	3	2	2	2	3	2	2	3	3	1
External Component Failure	4	3	5	2	4	1	5	2	5	1
Performance Shortfalls	3	2	3	1	4	3	3	3	4	1
Organizational Risk	2	1	1	3	2	3	3	4	4	3
Technical Integration Risk	4	3	3	4	3	3	5	2	5	3