RMMM Plan for USMS (University Society Management System)

A risk management technique is used in this software project to handle potential challenges that could impact the success of USMS. The RMMM strategy focuses on identifying risks, preparing for them through mitigation, tracking their evolution, and managing their outcomes effectively.

Risk Mitigation

To proactively reduce or eliminate risks in the USMS project, the following steps have been outlined:

Risk	Mitigation Strategy
Incomplete Requirements or Misunderstanding	Conduct multiple requirement-gathering sessions with end users (society heads, university admin, etc.). Validate requirements through mockups and feedback loops
Technology Stack Incompatibility	Select well-supported, familiar, and tested technologies (e.g., PHP, MySQL). Perform early prototyping to verify compatibility.
Data Loss or Corruption	Implement regular database backups. Use database transactions and validations. Version control for code.
Security Vulnerabilities	Apply authentication and authorization (e.g., admin vs. student roles). Use input validation, hashing, and secure login mechanisms. Conduct code reviews focused on security.
Resource Unavailability (team member absence)	Assign Backup responsibilities maintain shared documentation and code repositories,hold knowledge sharing sessions
Schedule Delays due to scope creep or external dependencies	Define and freeze scope early with stakeholders. Break project into sprints. Use agile boards to monitor progress.
Usability issues (non-intuitive interface)	Follow UI/UX principles. Create prototypes using Figma and conduct usability testing.Gather regular feedback from student users.
Poor System Integration (e.g., multiple modules fail to sync)	Develop modules with clearly defined APIs/interfaces, Use integration testing for early detection.

Risk Monitoring

During development, these activities will help track risks and their potential impact on the project:

Risk	Monitoring Activities
Incomplete Requirements	Conduct milestone reviews and walkthroughs with stakeholders. Compare features with documented requirements.
Technology Issues	Monitor integration challenges during developmentSTrack error logs, performance metrics, and user feedback.
Security Threats	Periodic vulnerability testing, code audits. Monitor login attempts and suspicious activities
Team Resource Issues	Track attendance, workload distribution, and morale during stand-up meetings.
Scope Management	Use a task management system (e.g., Trello) to monitor task additions or changes and their justifications.
UI/UX Feedback	Regularly collect feedback from test users; update design iterations as needed.

Risk Management and Planning

Assuming mitigation fails and risks become reality, the project manager will execute the following contingency plans:

Risk	Risk Management Strategy
Requirements Misalignment	Prioritize features based on impact. Conduct emergency requirement workshops to realign development with current needs.
Module Failure	Focus team efforts on critical modules first. Delay low-priority modules until stable version is achieved.
Key Team Member Dropout	Redistribute tasks. Use shared documentation and Git history to onboard new members quickly. Extend deadlines if needed.
Critical Bug or Security Breach	Immediately rollback to last backup. Notify stakeholders. Patch and release fixed version after testing.
UI Rejection by Users	Rollback to previous working design. Use emergency user survey to redesign with critical improvements.
Integration Failure between Modules	Conduct emergency code merge with both frontend and backend teams. Revisit API structure. Pair programming may be used

This RMMM plan will evolve throughout the project. Updates will be made at major milestones or if new risks are discovered. Documentation and tracking will be handled via shared repositories and project boards to maintain transparency and preparedness.