Risk Management and Configuration Management Exercise for Cinevea

Simulated Project Scenario

Project: Develop a web-based application for personalized movie recommendations and review aggregation.

Key Features:

1. AI-driven personalized movie recommendations.
2. Aggregated reviews from platforms like IMDb, Rotten Tomatoes, and Metacritic.
3. Interactive chatbot for user engagement.
4. User-generated content for community interaction.

Project Duration: 6 months
Team: 6 members (2 developers, 1 tester, 1 UX designer, 1 project manager, 1 business analyst)

---

1. Risk Management Plan

Template

1. **Objective** Identify, assess, and mitigate risks to ensure the successful delivery of Cinevea.
2. **Risk Identification** Identify potential risks throughout the project lifecycle.
3. **Risk Assessment** Assess risks based on their likelihood and impact.
4. **Risk Mitigation Plan** Develop strategies to minimize or eliminate risks.
5. **Monitoring and Reporting** Continuously monitor identified risks and report their status.

Exercise

Risk Identification

Risk ID	Risk Description	Category
R1	Dependency on external APIs (IMDb, Rotten Tomatoes, Metacritic).	Technical
R2	Performance issues under high user load.	Performance
R3	Security vulnerabilities related to user data storage and authentication.	Security
R4	Regulatory compliance challenges (GDPR, data privacy laws).	Legal

R5	Mismanagement of user-generated content.	Content Moderation
R6	Competitive pressure from established platforms.	Market
R7	Complexity due to AI integration and real-time chatbot.	Development
R8	Potential delays in feature deployment due to iterative risk identification.	Project Management

Risk Assessment

Risk ID	Likelihood	Impact	Priority
R1	High	High	Critical
R2	Medium	High	High
R3	High	High	Critical
R4	Medium	Medium	Medium
R5	High	Medium	High
R6	Medium	High	High
R7	High	Medium	High
R8	Medium	Medium	Medium

Risk Mitigation Plan

Risk ID	Mitigation Strategy
R1	Implement fallback mechanisms for API failures; establish multiple data sources.
R2	Optimize backend infrastructure for scalability; conduct load testing.
R3	Enforce encryption and multi-factor authentication; conduct regular security audits.
R4	Monitor regulatory updates and ensure compliance.
R5	Implement automated moderation and community reporting tools.
R6	Differentiate through unique features and strategic partnerships.
R7	Adopt modular architecture for easier integration and maintenance.
R8	Allocate buffer time for feature rollouts; conduct regular risk reviews.

Monitoring and Reporting

- Weekly risk review meetings.
- Update the risk register on GitHub as issues are logged or resolved.

---

2. Configuration Management Plan

Template

1. **Objective** Manage and track changes to project artifacts and ensure consistency throughout the project lifecycle.
2. **Configuration Identification** Define artifacts to be managed (e.g., source code, documentation, test cases).
3. **Version Control** Use GitHub for versioning, branching, and collaboration.
4. **Change Control** Establish a process for submitting, reviewing, and approving changes.
5. **Configuration Audits** Conduct periodic audits to verify compliance with configuration policies.

Exercise

Configuration Identification

Artifact	Owner	Tool
Source Code	Developers	GitHub
Project Documentation	Business Analyst	GitHub Wiki
Test Cases	Tester	GitHub Issues
UX Designs	UX Designer	Figma (linked in GitHub)

Version Control

Repository Structure:

```
/Movie-Recommendation
  /src
  /docs
  /tests
  /models
  /datasets
  /designs
```

Branching Strategy:

- `main`: Production-ready code.
- `dev`: Active development branch.
- `feature/*`: Individual features.

Commit Message Guidelines:

- `feat:` Adding a new feature.
- `fix:` Fixing a bug.
- `docs:` Changes to documentation.
- `perf:` Performance improvement.

Change Control

1. **Request a Change:** Submit a GitHub issue using a predefined template.
2. **Review and Approval:** Assign reviewers using GitHub pull requests.
3. **Merge Changes:** Only merge after passing code reviews and CI tests.

Configuration Audits

- Schedule bi-weekly reviews for:
  - Branch hygiene.
  - Consistent commit message usage.
  - Open vs. resolved issues.

Implementation Using GitHub

1. **Create a GitHub Repository:**
   - Repository name: `cinevea`.
   - Add a README.md for project overview.
   - Add `.github/ISSUE_TEMPLATE` for consistent issue reporting.
2. **Set Up a Project Board:**
   - Columns: To Do, In Progress, Review, Done.
3. **Automate Workflows:**
   - Use GitHub Actions for CI/CD pipeline (e.g., run tests on pull requests).
4. **Risk Register:**
   - Use a Markdown file (e.g., `RISK_REGISTER.md`) to document risks.
5. **Track Changes:**
   - Use GitHub's issue tracking and pull request features for all configuration items.

By following this structured approach, Cinevea ensures streamlined risk management and effective configuration management, enhancing system reliability, user trust, and platform performance.