**Configuration Management Plan for YourFinance**

**1. Objective**

The objective of this Configuration Management Plan is to ensure that all project artifacts related to ***YourFinance*** are effectively managed, version-controlled, and tracked. *YourFinance* is a web-based personal finance management application that includes:

* User authentication and secure access.
* Financial data input (salary, investments, liabilities, expenses).
* AI-powered financial advice and goal tracking.
* Financial dashboard for data visualization.
* Integration with third-party financial APIs (e.g., banking, tax services).
* Data security and encryption.

Effective configuration management ensures collaboration, consistency, security, and project stability throughout the software development lifecycle.

**2. Configuration Identification**

Configuration identification helps define and track key project components, ensuring that all updates and changes are documented systematically.

|  |  |  |
| --- | --- | --- |
| **Artifact** | **Owner** | **Tool** |
| **Source Code** | Developers | GitHub |
| **Project Documentation** | Business Analyst | GitHub Wiki |
| **Test Cases** | Tester | GitHub Issues |
| **UI/UX Designs** | UX Designer | Figma (linked in GitHub) |
| **CI/CD Pipeline Configurations** | DevOps Engineer | GitHub Actions |
| **API Specifications** | Developers | GitHub & Postman |

Each artifact has a dedicated repository section with proper versioning and access controls to prevent unauthorized modifications.

**3. Version Control & Change Control**

Version control ensures that project changes are systematically recorded, reviewed, and approved before implementation.

**3.1 GitHub Repository Structure**

A well-organized repository structure is maintained to manage different components efficiently:

/yourfinance

/src # Source code (backend & frontend)

/docs # Project documentation (SRS, API docs, user manuals)

/tests # Test scripts and test case documents

/designs # UX/UI wireframes, mockups

/configs # CI/CD, environment, and API configuration files

**3.2 Branching Strategy**

* **main**: Stable, production-ready branch.
* **dev**: Active development branch.
* **feature/**\*: Individual feature development branches (e.g., feature/ai-advice).
* **hotfix/**\*: Used for urgent fixes on production.

**3.3 Commit Message Guidelines**

Following Conventional Commits ensures clarity in tracking changes:

* feature: Added financial dashboard enhancement.
* fix: Resolved API authentication issue.
* docs: Updated banking API integration guide.
* test: Added unit tests for Ai chatbot model.

**3.4 Change Request Process**

To ensure controlled modifications, the following steps must be followed:

1. **Request Change**: Submit a GitHub issue with a detailed description.
2. **Review & Approval**: Assign reviewers and conduct code reviews via GitHub Pull Requests.
3. **Merge & Test**: Merge changes only after passing CI/CD tests and security audits.
4. **Deploy & Monitor**: Monitor the application post-deployment for any failures.

**4. Configuration Audits**

To maintain consistency, security, and reliability, configuration audits are performed at bi-weekly intervals. The following areas are reviewed:

* **Branch Hygiene**: Ensuring unused branches are merged and deleted.
* **Commit Consistency**: Verifying adherence to commit message guidelines.
* **Access Control**: Reviewing repository permissions to restrict unauthorized changes.
* **Open vs. Resolved Issues**: Ensuring timely resolution of issues to prevent project bottlenecks.
* **Dependency Management**: Checking for outdated libraries and security vulnerabilities.

These audits are logged in GitHub Wiki for future reference and team transparency.

**5. Implementation Using GitHub**

A structured GitHub-based Configuration Management System is established:

**GitHub Repository Setup**

* **Repository Name**: yourfinance
* Protected branches (main, dev) to prevent accidental changes.

**README.md Setup**

* Clearly documents project objectives, setup instructions, and contribution guidelines.

**Project Board for Task Management**

* Columns: To Do, In Progress, Code Review, Done
* Tasks assigned using GitHub Issues & linked pull requests.

**Automated CI/CD Integration**

* **GitHub Actions** automates tests and deployments.
* Every pull request triggers:
  + Unit tests (JavaScript, Python, etc.)
  + Security scans (OWASP ZAP, Snyk)
  + Code linting (ESLint, Prettier)
  + Deployment to staging environment

**Risk Register Maintenance**

* Risk-related issues tracked in RISK\_REGISTER.md.
* Assigned risk owners responsible for mitigation actions.

**Issue Tracking & Pull Request Workflow**

* All changes logged as GitHub issues.
* PRs require at least one approval before merging.
* CI/CD must pass before deployment.

**6. Monitoring & Reporting**

To ensure continuous tracking and governance, the following monitoring methods are in place:

* **Weekly Code Review Meetings**: Review ongoing PRs and development updates.
* **Audit Logs for Security & Compliance**: All repository activity is logged for compliance tracking.
* **CI/CD Pipeline Reports**: Monitors test success rates, failed deployments, and bug frequency.
* **Versioning & Release Notes**: All feature releases include detailed change logs.

**7. Configuration Management Challenges & Mitigation**

|  |  |
| --- | --- |
| **Challenge** | **Mitigation Strategy** |
| **Code Conflicts** | Encourage frequent merging, enforce PR reviews, use feature toggles. |
| **Unauthorized Changes** | Implement GitHub branch protections and role-based access control. |
| **Versioning Issues** | Follow strict semantic versioning guidelines and maintain release notes. |
| **Documentation Gaps** | Use automated documentation tools like Docusaurus and Swagger. |
| **CI/CD Failures** | Implement rollback strategies and blue-green deployments. |

**8. Conclusion**

By implementing this Configuration Management Plan, *YourFinance* ensures:

* Robust version control through structured GitHub workflows.
* Efficient change tracking via pull requests and issue logs.
* Secure software development using CI/CD automation & access control.
* Transparent collaboration with audit logs and structured documentation.

This structured approach guarantees stability, compliance, and scalability, leading to a high-quality financial management platform.