Project Report: Configuration Management and Procurement Process

1. Introduction

This report details the configuration management and procurement processes implemented for the **Procurement Management System** project. It includes a description of the branching strategy, version control practices, deployment configurations, and the tools and services acquired for the system's successful delivery.

2. Configuration Management

2.1 Version Control

The project utilizes **Git** for version control, hosted on **GitHub**. The following practices were adopted:

- **Main Branch**: Represents the production-ready code. All stable releases are merged into this branch.
- **Feature Branches**: Each new feature is developed in its own branch, named after the feature being developed (e.g., `feature-add-request`, `feature-edit-request`).
- **Pull Requests**: Features are merged into the main branch via pull requests, ensuring proper code review and testing before deployment.

2.2 Branching Strategy

We adopted a **GitFlow** workflow for managing branches and releases:

- **Feature Branches**: All new features were developed in isolated feature branches to minimize disruption to the main codebase.
- **Development Branch**: Once features were merged, they were tested in a development branch.
- **Hotfix Branches**: In case of bugs found in the main branch, hotfixes were developed separately and merged back into both main and development branches.

2.3 Deployment Configuration

Deployment was managed using automated scripts that handle:

- **Environment Setup**: The `.env` file contains environment-specific configurations, such as `FLASK_ENV`, `SECRET_KEY`, and database URIs.
- **Database Migrations**: **Flask-Migrate** was used to handle database schema changes.
- **Production Deployment**: The system was deployed to a cloud-based hosting provider that supports **Flask** applications (e.g., AWS or Heroku).

2.4 Version Control Tools

The following tools were utilized for version control and configuration management:

- **Git**: For tracking changes in the codebase.

- **GitHub**: For hosting the Git repository and managing issues and pull requests.
- **GitHub Actions**: For Continuous Integration (CI) to run tests on every commit and pull request.

3. Procurement Management

3.1 Tools and Services Acquired

Several tools and services were identified and acquired to ensure the success of the project. These include:

- **Cloud Hosting**: A scalable cloud hosting service was chosen to ensure reliability and availability.
- **Relational Database**: A PostgreSQL database was selected for its performance and integration with Flask.
- **Version Control**: GitHub was chosen to host the code repository and manage collaboration.

3.2 Procurement Process

The procurement process involved the following steps:

- **Identification of Needs**: Based on the project requirements, key services such as hosting, database, and development tools were identified.
- **Supplier Research**: Several suppliers were evaluated based on cost, service quality, and ease of integration.
- **Request for Proposals (RFP)**: An RFP was sent out to selected suppliers, and proposals were reviewed.
- **Final Selection**: Suppliers were selected based on a balance of cost, service reliability, and technical support.

4. Conclusion

The **Procurement Management System** project successfully implemented both configuration management and procurement strategies. The use of a GitFlow branching strategy ensured a smooth development process, while the procurement of reliable hosting and database services helped to ensure the system's scalability and robustness. The project was delivered on time and within budget, meeting all functional and non-functional requirements.