**Software Development Reflection Essay**

**Project Overview**

The project focused on creating a service request management system (SMS) for a fictional company called TechPro Solutions. I developed the system using Python and SQLite, utilizing software engineering techniques and object-oriented programming (OOP) principles to mimic real-life enterprise support operations. The application is designed to cater to three different roles: administrator, support engineer, and customer, with each role having its own set of privileges and responsibilities..

**Design Process**

I began with a clear functional specification, outlining required features such as:

* Role-based access control
* Ticket creation, assignment, and status tracking
* Basic notifications through ticket history updates
* Command-line user interface (CLI)
* Lightweight local data storage using SQLite

The application architecture was broken into three modules:

* auth.py for authentication logic
* ticket.py encapsulating all ticket management logic via the TicketSystem class
* main.py serving as the entry point and menu controller

Object-oriented design was used to encapsulate user roles and ticket behaviors, promoting modularity, code reuse, and ease of maintenance.

**Implementation Highlights**

Key programming practices were followed:

* **Encapsulation:** All ticket operations are managed inside the TicketSystem class.
* **Separation of Concerns:** Authentication, main flow, and ticket operations were kept in distinct modules.
* **SQL parameterization:** Prevented SQL injection by using parameterized queries.
* Error handling: Included basic error management for invalid input or unauthorized access.
* **Data persistence:** Implemented using SQLite, offering simplicity and portability without external dependencies.

**Challenges Encountered**

1. **Managing Role-based Menus**: Designing a logical framework for three separate roles while avoiding code duplication demanded careful consideration.
2. **Ticket History Tracking:** Using a semicolon-delimited string as a basic method for notifications and logging was straightforward, but it demanded a strong commitment to keeping the content clear and readable.
3. **Database Integrity**: It was crucial to ensure that user IDs, usernames, and roles were consistently connected throughout all operations.
4. **Testing CLI Logic:** Testing CLI applications can often be a manual process, especially without a graphical user interface or REST API. To address this challenge, I developed unit tests that sidestepped the need for CLI input wherever we could.

**Testing and Validation**

I validated the system with:

* Manual testing for each user role and feature
* Automated unit tests for login, ticket creation, and updates
* A sample database (test\_data.db) with predefined users and tickets

The application passed all expected functional cases, proving its reliability for basic enterprise workflows.

**Skills and Knowledge Gained**

Through this project, the following skills were enhanced:

* Applying object-oriented principles in a multi-user context
* Using SQLite for data modeling and storage
* Writing modular, testable Python code
* Building CLI applications with logical menu navigation
* Collaborating with Git and GitHub for version control and backup

**Future Improvements**

If further developed, the system could benefit from:

1. GUI Interface using Tkinter or PyQt to enhance usability.
2. Advanced ticket filtering (by status, engineer, or date).
3. Email/SMS notifications upon ticket updates or assignments.
4. Role-based dashboards summarizing key metrics for Admin and Engineers.
5. Multi-language support for international use.
6. Security enhancements, such as password hashing and user session tracking.

**Conclusion**

This project successfully integrated theory with practical application in software development. It highlighted the significance of structured thinking, modular coding, and user role modeling, illustrating how even simple tools like SQLite can contribute to building effective enterprise solutions. Not only did it achieve its functional goals, but it also laid a strong groundwork for future improvements and potential deployment in a real-world small business context.