# CHAPTER V
# CONCLUSION AND RECOMMENDATIONS

## 5.1 Conclusion

Based on the results of the analysis, design, implementation, and testing detailed in the preceding chapters regarding the Academic Information System (E-Learning) at SMA Negeri Jatirogo, the following conclusions can be drawn:

1. The application of the Prototyping methodology proved effective in mapping and realizing the operational needs of SMAN Jatirogo. Through iteration cycles involving users directly, the system successfully implemented all eight Must Have category features, encompassing Hybrid Grading automation, material distribution centralization, and CSV-based mass data management. This success is empirically validated through User Acceptance Testing (UAT), which recorded a 100% compliance rate from all involved actors, proving that the built platform fully meets the actual user needs in the field.
2. The implementation of encapsulation and bounded context principles from Domain-Driven Design (DDD) was successfully applied to the Modular Monolith architecture by partitioning the system into four autonomous domains: Auth, Admin, Course, and Quiz. Each module has isolated internal layers (core, infrastructure, repositories) and communicates only through public interfaces (contracts), meaning internal changes in one module do not affect other modules. The successful implementation of this principle is empirically proven via structural testing WB-01, confirming zero cross-module communication violations, and WB-02, validating zero layering direction violations. These results prove the system has low coupling and high modularity, as intended by this study's architectural design.
3. System feasibility evaluation was conducted through two complementary testing mechanisms. From the functional side, Black Box testing using Equivalence Partitioning (EP) and Boundary Value Analysis (BVA) methods across 15 testing scenarios yielded a 100% Valid success rate, while also proving the system's robustness in neutralizing input anomalies

without causing system failure. From the structural side, White Box testing via Cyclomatic Complexity analysis proved that all critical functions operate within reasonable tolerance thresholds without any entering the high-risk category. The synergy between functional and structural testing confirms that the SMAN Jatirogo e-learning system has met the feasibility standards from two different software quality dimensions: external behavioral correctness and internal logic integrity.

## 5.2 Recommendations

Although this system is operational and meets current requirement specifications, there is still room for development to perfect the digital ecosystem at SMAN Jatirogo. The recommendations that can be provided for future research or development are:

1. Mobile App-Based Interface Development: Considering the majority of students access learning materials via smartphones, developing a native mobile application (e.g., using React Native or Flutter) is highly recommended. Thanks to the separation of public interfaces (contracts) already prepared in the current Modular Monolith architecture, mobile applications can directly consume existing APIs (Application Programming Interfaces) without needing to overhaul the server logic (backend).
2. Artificial Intelligence Integration for Hybrid Grading: The Hybrid Grading feature currently still requires teacher intervention to manually assess essay parameters. Future research can explore the implementation of Machine Learning or Natural Language Processing (NLP) models to automatically correct and assign point weights to essays based on reference keywords, to further lighten educators' administrative burdens.
3. Escalation to Microservices Architecture: If there is an exponential escalation in the number of users in the future (e.g., the system being used by all high schools in the Tuban Regency area) that burdens traffic on a single server, the currently isolated module structures (such as the Quiz Module) can be extracted and orchestrated into independent microservices using containers (such as Docker and Kubernetes).