CSC505 Portfolio Project - Automated Teller Machine (ATM)

# 1. Lessons Learned Reflection – CSC505: Principles of Software Development

Throughout CSC505: Principles of Software Development, I have acquired a foundational and practical understanding of how software systems are conceived, designed, implemented, and maintained. One of the most significant lessons I’ve learned is the value of using software models, such as UML diagrams, to translate abstract ideas into tangible system representations. These models are not only helpful for organizing thought processes but are also indispensable tools for communicating design to other stakeholders, including developers, project managers, and clients.  
  
The course emphasized how algorithms and models are used to approach real-world software problems. In my Portfolio Project, for instance, creating a UML state diagram for an ATM system required breaking down a seemingly simple process—authentication, transaction handling, and state transitions—into clearly defined states and events. This exercise sharpened my ability to think modularly and see software not just as code, but as a system with defined behavior.  
  
Another major area of growth for me was in identifying and using basic data types, and recognizing how control structures—conditional statements and loops—are used in nearly every software application. These are the foundational building blocks of programming, and mastering them has enabled me to write more structured and maintainable code. Implementing the ATM script using conditionals and counters helped reinforce how theoretical concepts map to practical implementations.  
  
Moreover, the course gave me practical experience in using appropriate data structures such as lists and dictionaries to manage data flow and operations. For example, storing transaction logs or tracking login attempts would be difficult without understanding how to structure and access data efficiently. This directly aligns with Course Learning Outcome 1.3.  
  
In learning to compare and apply software development models like Waterfall and Agile, I now appreciate that the right model depends on the project type, team structure, and stakeholder requirements. In class discussions and assignments, we analyzed the pros and cons of each, giving me clarity on when and why one approach might be preferred over another. This understanding aligns with CLOs 1.4 and 1.5 and will be especially helpful in project planning and stakeholder communication.  
  
This course has also helped me understand why software takes so long to finish and why maintenance is a never-ending task. Requirements evolve, systems become complex, and users have unpredictable needs. Through critical thinking exercises, I’ve learned that flexibility, communication, and planning are just as important as coding.  
  
On a personal level, this course has made me more detail-oriented and confident in system design. I now regularly think in terms of state transitions, data flow, and failure conditions before writing any code. Professionally, it has influenced how I approach technical problem-solving with my team—I'm better equipped to evaluate trade-offs, model solutions, and communicate more effectively.  
  
In conclusion, CSC505 has offered a comprehensive roadmap from theory to practice. Whether I’m building a new system from scratch or improving an existing one, the lessons I’ve learned—from modeling and development to analysis and reflection—are now a core part of how I work as a software engineer.

# 2. UML State Diagram Description

The UML diagram created for this project is a state diagram for an ATM system. It illustrates the states involved when a customer attempts to withdraw money. The flow includes authentication using a PIN, tracking incorrect attempts, handling exceeded limits, and closing the account if the balance is zero.

A diagram of a payment method

AI-generated content may be incorrect.

Entry and exit actions are used where necessary to model transitions to the same state or upon state entry.

# 3. Python Script Explanation

The accompanying Python script simulates the ATM workflow as described in the UML diagram. It allows for PIN verification, limits login attempts, and handles transactions based on account balance. The script prints each step in sequence, simulating a real-world ATM session.



# 4. Execution

Enter Pin

A screen shot of a computer program

AI-generated content may be incorrect.

Auth Success

A screenshot of a computer program

AI-generated content may be incorrect.

Check balance

A screenshot of a computer

AI-generated content may be incorrect.

Withdraw 20$

A screenshot of a computer

AI-generated content may be incorrect.

Exit

A screenshot of a computer

AI-generated content may be incorrect.

# 5. References

- Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.  
- Interaction Design Foundation. (2016). User Interface Design. https://www.interaction-design.org/literature/topics/ui-design