Developing UML Diagrams for

Master of Science in Information Technology

Software Design and Programming

Joseph DiBiasi, Estell Moore, James McKenna, Anthony Martuscelli

University of Denver College of Professional Studies

10/28/2025

Faculty: Nirav Shah, M.S.

Director: Cathie Wilson, M.S.

Dean: Michael J. McGuire, MLS

Abstract

In a professional and class setting, visualizing and developing software or architecture for various problems can lead to difficulties or harder issues in the future. UML Diagrams can help not only visualize those problems but also help with creating software and infrastructure for the scenarios you're trying to approach. In our final team assignment, we're developing UML Diagrams that can not only represent our Vendors, Shop Keepers, and all the information they need to retain; but also showcase this in a way to show extensibility and maintainability through our classes and their relationships to one another. This will be demonstrated by our diagrams in our appendices and our challenges and design decisions we faced along the way.

CONTENTS

Introduction

UML diagrams are an important part of the Software Development Life Cycle. While written use cases are a good way of laying out the details of how we want our users to interact with our systems, it is the UML Diagrams that help bring those details to life. They do this by taking the actors and flows defined within written use cases and visually displaying that information as a connected series of interactions. This essentially bridges the gap between those with programming knowledge and those without, keeping both informed of the requirements, processing, and functionality desired by a system. UML diagrams come in several forms, and our goal was to build on the scenario three use cases defined by our group in week one by developing corresponding activity, sequence, and interaction diagrams.

Design Strategies

For design, we wanted to keep conventional UML formatting, while ensuring maximum readability in our diagrams. First priority was to understand exactly what we were tasked with. The task was to aid a shopkeeper with inventory and ordering, specifically when interacting with vendors. We used the case overview as the foundation for our design. This helped identify the main functions in the system such as the vendor records, contact info, rep data, and communication history.

We started with defining the primary actor, the Shopkeeper. This will be the primary user of the system. Taking in the shopkeeper perspective helped guide us with primary use cases such as adding vendors, updating vendor information, and responses. Throughout this process, we were able to identify secondary actors via the flow of events. Being able to see

these secondary actors allowed us to construct our design in a manner that reflected real-world dependencies and interactions relevant to actual vendor management scenarios.

Alternative flows within our use cases dictated the inclusion of fragments into our UML. For example, exception handling events such as “Vendor not Found” were modeled with alternative paths to promote a more resilient system.

Boosts and Challenges

The most significant challenge in creating the UML diagrams was overcoming the learning curve associated with the available design tools. The initial difficulty lay in identifying a reliable platform to construct the diagrams. Several strong options were considered such as PlantUML, Lucidchart, and various templates provided in earlier assignments. Ultimately, to maintain simplicity and consistency, Lucidchart was chosen for the development of all three diagrams. Then, finding a template or appropriate tool area to utilize for the respective charts was difficult.

One of the main areas of concern from the assignment was the “alternative” blocks in the sequence diagram. To apply them effectively, additional research was necessary to understand how to structure and populate these sections correctly. Another area of difficulty was identifying and defining the appropriate components of the diagram. However, once those components were established, maintaining a simple and general layout became much more manageable.

The activity diagram was likely the simplest of the three. It provided a clear depiction of the software's workflow and control flow from the shopkeeper's perspective, while incorporating the previously defined use cases. Although designed to remain straightforward, the main challenge was stepping back to simplify the software's expected activities rather than focusing on its internal settings and configurations.

The interaction diagram, in contrast, explored the system's internal components and how they communicate during specific task executions. The primary difficulty with this diagram was maintaining simplicity while accurately identifying the expected interactions and the appropriate setters within the code.

Conclusion

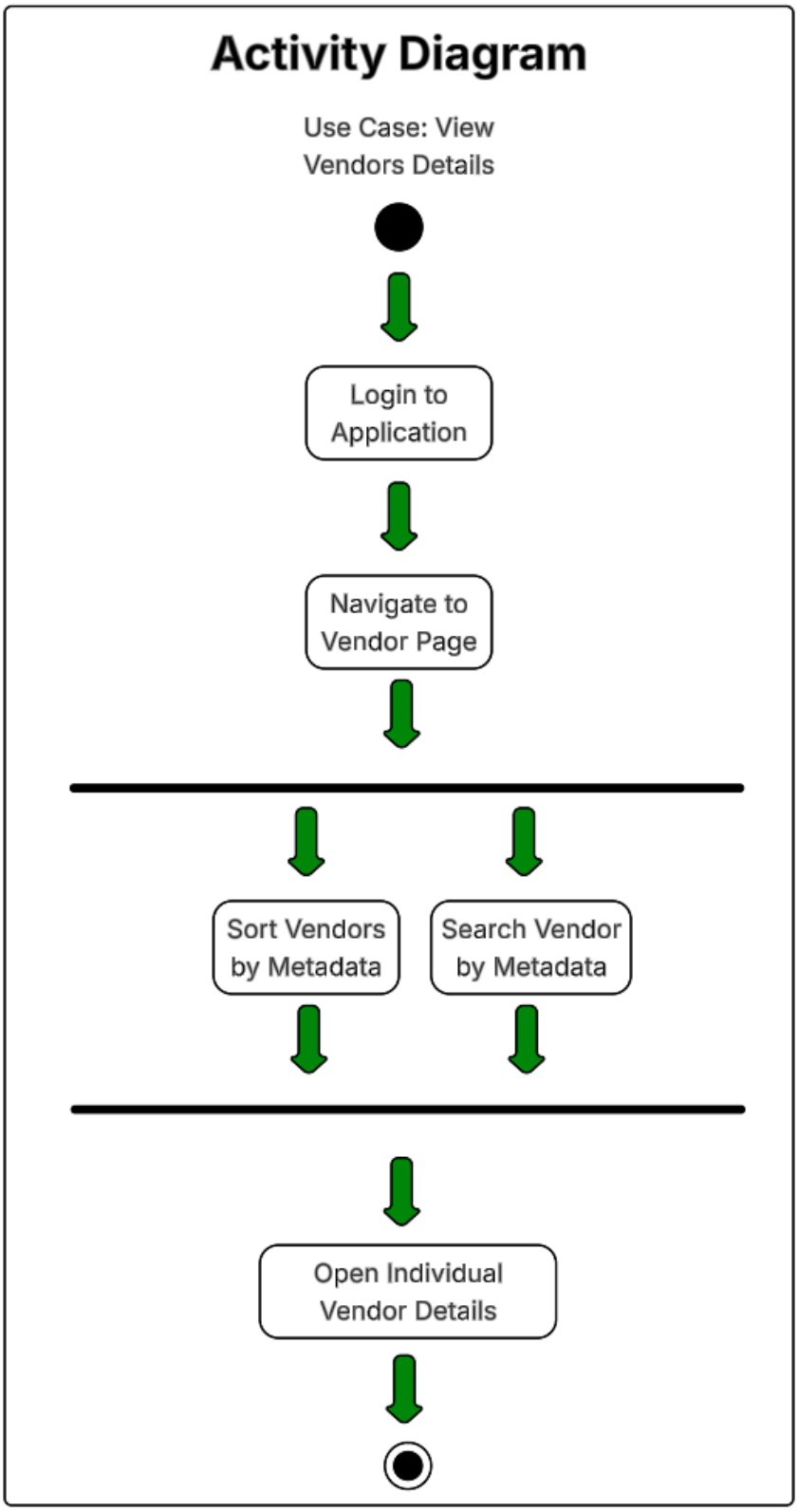
As stated in our primary objective to have an object oriented design that would capture our structure for our maintainable and extensible code. It was a desire to use class hierarchies to show the interaction, extension, and inheritance of how our classes not only reused code from one another but to demonstrate it in a visual way. Our class designs and the relationships between them have been to provide a system for our shopkeeper to have utilization of a tool that would be easier to modify and implement because of the UML Diagrams we have made. UML Diagrams allowed us to achieve this by showing missing relationships or effective ways one class could inherit or benefit another. Other iterations or future enhancements can be done on our system that would be easier to follow due to how our diagrams are designed and implemented into one another. Lastly, this also applies to future software projects and professional environments. UML Diagrams have made it easier to break down more complex

and difficult problems into bite sized chunks that can allow for easier software design and modeling of capabilities.

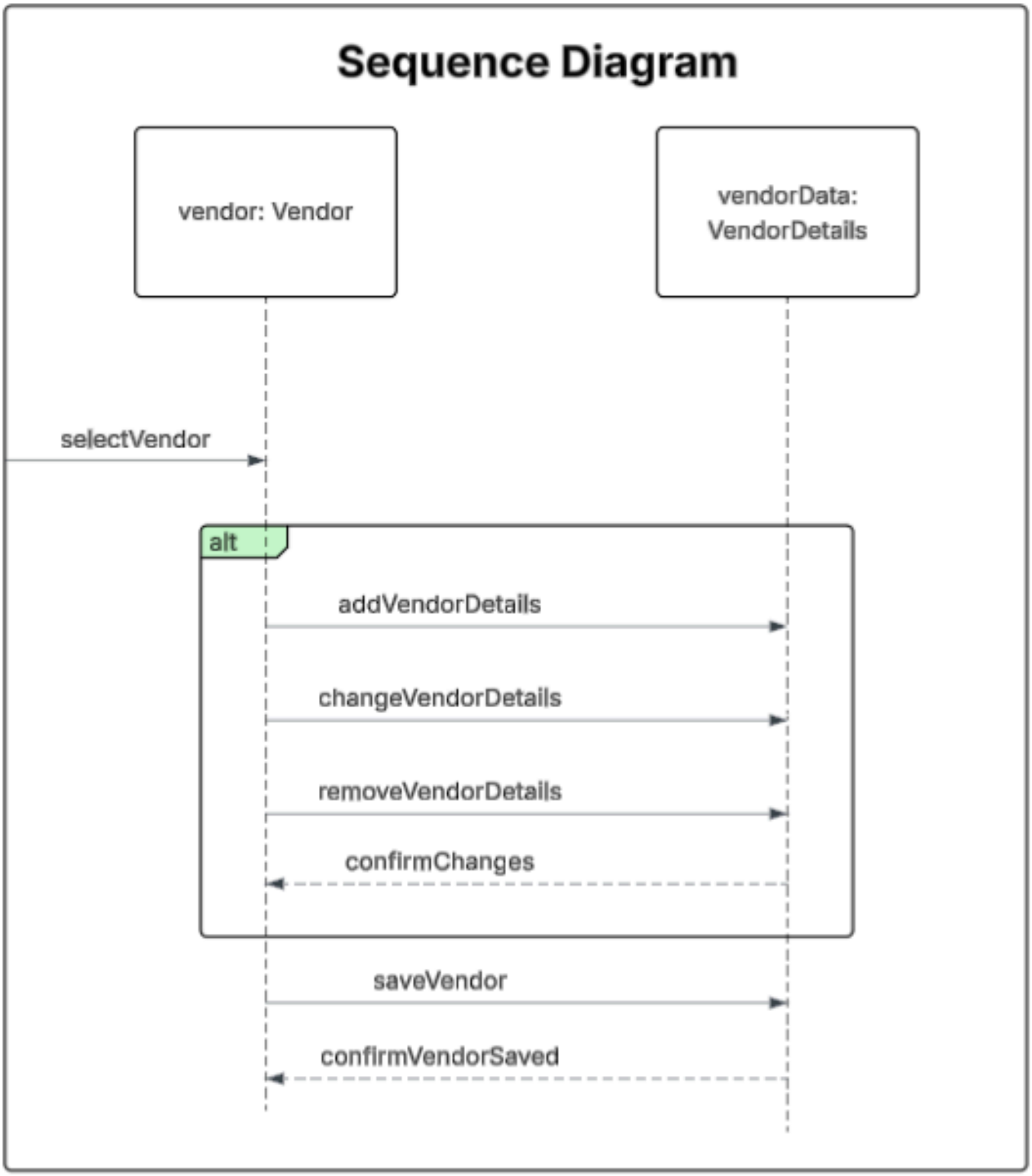
References

Jain, Ankit. 2025. “Unified Modeling Language (UML) Diagrams.” *GeeksforGeeks*. Last modified August 28, 2025. https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-introduction/.

Appendix A: Activity Diagram



Appendix B: Sequence Diagram



Appendix C: Interaction Diagram