COMP 200

Summer Practice Report

Umut Kadıoğlu
042201104

07.07.2025- 15.08.2025, 29 Workdays
Submitted: *28.09.2025*

MEF University

Computer Engineering Program

Executive Summary

Lumnion Bilişim Teknolojileri is a software company operating in the InsurTech sector, focusing on developing actuarial and artificial intelligence–based solutions for the insurance industry. The main engineering activities of the company include backend development with ASP.NET Core and Entity Framework Core, database management with SQL Server, and the integration of actuarial models and AI algorithms into scalable software platforms used by insurance and reinsurance companies.

During my internship, I was part of the CashFlow development team. My activities progressed step by step: in the first two weeks, I mainly focused on self-learning through Udemy courses, observing senior developers, and handling small tasks such as minor fixes and migration checks. From the third week onwards, I started taking responsibility for backend tasks, including:

- Developing the Save Rule endpoint, merging create and update operations into a unified, more maintainable API.
- Implementing enumeration functions in SQL (e.g., INT_TO_ENUM, ENUM_SIZE, READ_FROM_TABLE) to support actuarial formulas.
- Integrating user-specific settings (delimiter, digit count, sample size, error thresholds) into backend services.
- Designing an endpoint to check Master Product–Indicator usage, ensuring relational integrity.
- Debugging and resolving frequent errors such as NullReferenceException and bulk insert issues.
-

My initial expectation was simply to improve my technical skills and gain exposure to real-world projects. However, the outcomes went beyond that: by the end of my internship, I was handling general backend engineering tasks on my own and directly contributing to features that improved system reliability. Daily stand-ups, sprint planning meetings, and weekly product/investor sessions also gave me insight into how a professional software team organizes its workflow.

Overall, I gained both technical and professional growth. I improved my coding skills in C#, SQL, and API design, learned to manage database migrations and error handling, and saw how engineering decisions directly affect business outcomes. Beyond the technical side, I learned the value of self-directed learning, teamwork across different roles, and even explored topics like ethical hacking with my senior mentor. This internship gave me not only practical experience but also the confidence and mindset required to continue developing as a software engineer.

Contents

1. Company and Sector (*maximum ten pages).*

a. Overview of the Company and Sector (*maximum two pages).*

Company Information

The company is Lumnion Bilişim Teknolojileri A.Ş., a privately held startup operating in the InsurTech sector Its full address is Ömer Avni Mahallesi, İnebolu Sokak, Beyoğlu/İstanbul, Türkiye, and its official website is https://www.lumnion.com. Founded with the mission of bridging actuarial science with artificial intelligence, Lumnion defines itself as an AI-Driven End-to-End Insurance Pricing Platform.

The company has approximately 19–20 employees, with no blue-collar workers. Mr. Eray leads the engineering division as Chief of Engineering, managing a team of nine engineers: four working on the Platform modules (Bee and Cheetah) and five on the CashFlow product.

The Platform

Lumnion's solutions are structured under a comprehensive Platform that brings together different modules supporting insurers at every stage of the pricing process. This end-to-end platform allows insurance companies to integrate data preparation, actuarial modeling, machine learning, rule engines, and reporting in one environment. It is designed to ensure transparency, regulatory compliance, and operational efficiency. (Lumnion, 2025)

Products and Modules

- **Bee:** A module focused on automated data preparation. It manages cleaning, transformation, and premium calculations, reducing manual work for actuaries.
- **Cheetah:** A module dedicated to risk pricing and actuarial modeling, supporting advanced statistical and machine learning methods such as GLM, GAM, XGBoost, and Random Forest. It ensures transparency by converting ML outputs into coefficients and base prices.
- **CashFlow:** A new product still in development (and the one I contributed to during my internship), designed to handle cash flow projections, actuarial modeling, and financial simulations for insurers.

Customers and Competitors

Lumnion serves both domestic and international clients, reflecting the global nature of the insurance technology sector. While the company has a strong presence in Turkey, it also competes in international markets against well-known players such as Prophet, SAS, and Guidewire. Unlike these larger corporations, Lumnion differentiates itself with agility, modular design, and AI-driven transparency.

Sector and Role

The company operates in the broader InsurTech / insurance software domain. This sector is rapidly expanding due to advances in big data, cloud solutions, and AI. Lumnion's role is to modernize risk pricing and insurance operations by combining actuarial expertise with machine learning and scalable backend systems.
In summary, Lumnion positions itself as both a platform provider and a modeling partner, delivering tools that improve accuracy, compliance, and efficiency in insurance decision-making.

b. Organization of the Company *(maximum two pages)*

Organizational Structure

Lumnion Bilişim Teknolojileri operates with a streamlined and specialized organizational structure that reflects the needs of a technology and software company. The structure is organic rather than mechanistic, meaning that communication between departments is highly flexible, decision-making is relatively decentralized, and teams collaborate closely to achieve project goals. This approach allows the company to adapt quickly to client needs and to the rapidly evolving insurance technology sector.

At the top of the hierarchy, the General Manager / CEO (Cenk Tabakoğlu**)** oversees the overall strategy and direction of the company. Reporting directly to the CEO are three main departments:

1. Technology and Development Department
Head: Chief of Engineering (Eray)
Team: 9 engineers divided into two product groups:
Platform Team (4 engineers): Working on Bee and Cheetah modules.
CashFlow Team (5 engineers): Developing the new CashFlow product.
Roles and responsibilities:
Software Engineers: Build and maintain back-end and front-end systems (.NET, SQL, APIs).
Data Scientists / Actuarial Specialists: Develop predictive models, risk pricing algorithms, and ensure compliance with actuarial principles.
Quality Assurance/Test Engineers: Responsible for debugging, testing, and ensuring product stability.

Collaboration: This department works directly with the analyst team to transform client requirements into software features.

2. Analysis and Client Relations Department
Head: Chief Analyst (Aren)
Team: 5–6 analysts.
Roles and responsibilities:
Conduct testing to ensure software aligns with actuarial and client requirements.
Act as a bridge between clients and engineers, translating customer feedback into technical tasks.
Support customer communication during and after implementation.

Collaboration: This group is critical for aligning technical outputs with client needs and ensuring that the platform remains user-focused.

3. Administration and Support Department
Head: Director (Tayfun)
Roles and responsibilities:

- Finance: Budgeting, payroll, and financial compliance.
- Human Resources: Recruitment, training, and employee development.
- Investor Relations: Maintaining transparent and regular communication with stakeholders and investors.

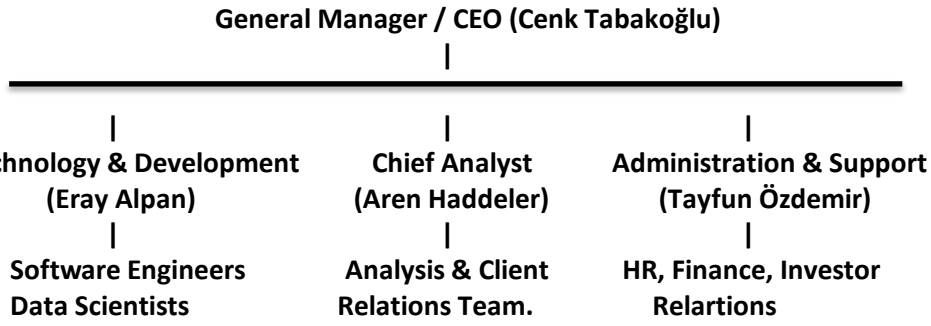
Interdepartmental Relationships

The Technology & Development team works closely with the Analyst team to ensure that client requirements are accurately implemented in the platform.

The Administration department supports both major teams by handling finance, HR, and investor communications.

The startup's organic structure ensures that even interns and junior engineers can directly collaborate with senior staff, while final approvals are made by department heads.

Organizational Chart (Textual Representation)



Conclusion

Overall, Lumnion's organizational structure reflects an organic and collaborative model. This structure fosters innovation, allows cross-functional teamwork, and ensures flexibility in responding to client needs, regulatory requirements, and technological advancements in the InsurTech sector.

c. Production/Service System *(maximum two pages)*

Service System Overview

As a software company, Lumnion Bilişim Teknolojileri does not operate a traditional manufacturing process. Instead, its "production" system is the development, testing, and deployment of actuarial and AI-based software solutions. This system is highly knowledge-intensive and requires collaboration between multiple departments. The service cycle integrates technology development, business development, finance, customer relations, and quality assurance into an iterative process that continuously delivers value to insurance companies.

Major Components of the Service System

1. **Marketing & Sales**
    - Identify market trends and client needs in the insurance sector.
    - Promote the Lumnion Platform and its modules (Bee, Cheetah, and CashFlow) to domestic and international clients.
    - Gather requirements and feedback to feed into the product backlog.
2. **Technology & Development**
    - Core responsibility for backend/frontend software engineering, actuarial modeling, and AI integration.
    - Tasks are managed through Agile methodology (daily stand-ups, bi-weekly sprint planning).

- Engineers work on separate GitHub branches, developing features independently.

3. **Finance & Administration**
   - Oversees budgeting, payroll, and licensing costs.
   - Allocates resources for R&D and ensures transparency for investors.
4. **Customer Relations & Support**
   - Provides technical support, client onboarding, and training.
   - Maintains continuous communication with clients to collect feedback.
5. **Purchasing & Logistics**
   - Ensures availability of cloud infrastructure (Microsoft Azure, AWS).
   - Manages tools, licenses, and external vendor relationships (e.g., SQL Server, GitHub).

Flow Chart of a Major Service (Software Development Cycle)

Customer Requirements

↓

Business Development & Analyst Team

↓

Technical Analysis & Sprint Planning

↓

Software Design (System Architecture)

↓

Coding & Implementation (Feature Branches in GitHub)

↓

Testing & Quality Assurance (Unit Tests, Integration Tests)

↓

Tester Approval & Code Review

↓

Merge into Release-Test Branch (Integration Environment)

↓

Deployment (Cloud / Customer Environment)

↓

Customer Training & Support

↓

Continuous Feedback (Investor, Product, and Client Meetings)

↓

Product Updates & Iterations

Productive Elements: feature development, actuarial modeling, testing, merging, deployment, training, and client support.

Non-Productive Elements: delays in tester approvals, sprint extensions (sometimes by one week), rework caused by unclear requirements, waiting for customer confirmation.

Performance of the Service System

The performance of Lumnion's service system is measured by:

Reliability: accuracy and stability of actuarial models and backend services.

Efficiency: delivering features within sprint timelines and reducing rework.

Customer Satisfaction: feedback gathered in weekly product meetings (Thursdays), investor meetings (Tuesdays), and customer onboarding sessions.

Scalability: ability to process larger datasets, comply with global standards (IFRS 17, Solvency II), and expand to new international markets.

Planning, Forecasting, and Logistics Activities

Planning: Managed through Agile. Each day begins with a daily stand-up at 9:00, and every two weeks a sprint planning meeting sets development priorities. Weekly product meetings on Thursdays track progress across teams, while investor meetings on Tuesdays align technical and financial expectations.

Forecasting: Market trends and regulatory changes are analyzed to predict demand for new actuarial features.

Inventory: Unlike physical manufacturing, inventory at Lumnion consists of cloud capacity, software licenses, and test datasets.

Logistics: Primarily digital ensuring cloud environments remain secure, test and production systems are synchronized, and deployment windows minimize customer downtime.

Conclusion

The production/service system at Lumnion is iterative, agile, and knowledge-based. Instead of physical goods, the company produces reliable actuarial software solutions. The use of GitHub branch-based workflows, strict tester approvals, and release-test integration ensures that only stable features reach the customer environment. Combined with continuous planning, forecasting, and feedback loops, this system enables Lumnion to adapt quickly to both technological advances and the evolving needs of the global insurance industry.

d. Professional and Ethical Responsibilities of Engineers *(minimum one page)*

Functions of Engineers in the Company

At Lumnion Bilişim Teknolojileri, engineers are at the core of the company's ability to deliver actuarial and AI-based solutions for the insurance industry. Their primary functions include:

- Designing and developing back-end services and APIs using C# and ASP.NET Core.
- Building actuarial and statistical models with algorithms such as GLM, GAM, Random Forest, and Gradient Boosting.
- Writing and optimizing SQL queries, managing database migrations, and integrating actuarial model data with financial datasets.
- Conducting debugging, unit testing, and error handling to ensure software quality.
- Collaborating with actuaries, analysts, and product managers to turn client requirements into technical features.
- Maintaining cloud infrastructures (Azure, AWS) for scalability, performance, and security.

Professional Responsibilities

The professional responsibilities of engineers at Lumnion extend beyond simply writing code. They are expected to:

- Deliver reliable, maintainable, and efficient code aligned with project requirements.
- Ensure actuarial models comply with Solvency II and IFRS 17 standards.
- Follow Agile practices—daily stand-ups, sprint planning, code reviews, and tester approvals.
- Document technical processes for traceability and team knowledge sharing.
- Respect the company's GitHub workflow, where no code can be merged into the release-test branch without senior and tester approval. This process guarantees accountability and technical integrity.

Ethical Responsibilities

Because Lumnion operates in the intersection of finance, insurance, and software, every technical decision carries ethical implications. The ethical responsibilities of engineers include:

- **Integrity and Honesty:** Results must be accurate and transparent. Manipulating models to produce misleading financial outcomes is strictly avoided.
- **Confidentiality:** Engineers protect sensitive financial and client data. During customer meetings, only senior staff (Alperen, Eray, Hüseyin, Aren) communicated directly, while others listened silently—an example of respecting information flow and confidentiality boundaries.
- **Accountability:** Every engineer is responsible for the quality of their work. If a feature fails in testing, it must be corrected before merging into the release branch. Responsibility is never shifted to others.
- **Fairness:** Since the software influences insurance pricing, models must be designed to avoid unfair discrimination and bias. This is a critical ethical requirement in InsurTech.
- **Sustainability:** Engineers are encouraged to write clean and efficient code that will remain maintainable for years, minimizing wasted time and computational resources.

Professional and Ethical Standards in the Company

Although Lumnion does not maintain a formal written Code of Ethics, ethical standards are embedded in its culture:

- IEEE/ACM Software Engineering Code of Ethics principles are followed indirectly, focusing on product quality, accountability, and public interest.
- Actuarial association guidelines influence how models are developed, ensuring they are transparent and not misleading.
- The company's culture is open, warm, and highly human-centric. Even though it is a small startup, ethical awareness is strong: from the most junior intern to the CEO, communication is direct and respectful. This environment itself functions as an unwritten but effective code of conduct.

Conclusion

From my observation, engineers at Lumnion balance technical competence with ethical awareness. Code cannot be pushed without senior review, ensuring accountability. Sensitive information is handled carefully, with only authorized voices communicating directly with clients. Although the company does not have a formal written ethics code, its people-oriented culture and professional practices ensure that ethical standards are upheld in every project. This balance of professionalism and ethical responsibility builds both the trust of clients and the integrity of the engineering team.

2. Summer Practice Description *(maximum fifteen pages)*

Overview of Activities

During my internship at Lumnion Bilişim Teknolojileri, I worked on several engineering tasks that combined backend development, database management, and API design. Some tasks were purely technical and required precision, while others taught me broader lessons about teamwork and communication in a professional environment. At times, I felt challenged—especially when debugging errors for hours—but these challenges became the most valuable learning opportunities.

Below is the main list of activities I performed, with explanations, challenges, and reflections.

1. Development of the Save Rule Endpoint

One of my first major assignments was to merge the Create and Update operations into a single Save Rule endpoint. This task was technically complex because it required restructuring existing code while ensuring data integrity.

- Technical Challenge: Handling null IDs and foreign key conflicts without breaking other parts of the system.
- Reflection: At the beginning, I was nervous about touching such a critical endpoint, but with guidance from senior engineers I gained confidence.
- Learning Outcome: I improved my knowledge of ASP.NET Core and Entity Framework Core, and learned how atomic transactions can prevent cascading errors.

2. Enumeration Functions in SQL

I worked on SQL-based functions such as INT_TO_ENUM, ENUM_SIZE, and READ_FROM_TABLE. These were essential for mapping categorical variables into actuarial models.

- Technical Challenge: The READ_FROM_TABLE function was especially difficult, since it required dynamic queries to handle variable row headers.
- Reflection: It was fascinating to see how abstract actuarial formulas could be translated into SQL logic. Debugging felt frustrating at times, but the moment it finally worked was very rewarding.
- Learning Outcome: I learned to design flexible database functions and gained appreciation for how enumeration supports complex business rules.

3. Entity Framework Core Migrations

I also worked on updating the database schema with new fields like Delimiter, Digit Count, and Sample Size.

- Technical Challenge: Migration errors such as invalid column names and locale mismatches often appeared during runtime.
- Reflection: These issues taught me patience—sometimes the solution was a single line of code, but it could take hours to discover.
- Learning Outcome: I understood how even small schema changes can impact the whole system and learned systematic debugging strategies in SQL Server.

4. User Settings Integration

Another key task was implementing user-defined parameters (delimiter type, digit count, sample size, and error thresholds). These directly affected how users interacted with the system.

- Technical Challenge: Ensuring smooth synchronization between frontend inputs and backend validation.
- Reflection: I enjoyed this task the most because I could immediately see its impact on usability. It reminded me that good engineering is not only about efficiency, but also about creating a positive user experience.
- Learning Outcome: I learned how configurable systems are designed and why usability is a critical engineering principle.

5. Master Product and Indicator Usage Check

I developed an endpoint to verify if indicators were already linked to a Master Product, preventing accidental deletion of active indicators.

- Technical Challenge: Writing optimized queries that fetched relational data quickly without affecting system performance.
- Reflection: At first, I underestimated how much this small feature could improve system reliability. When the frontend team confirmed that it reduced user errors, I realized its importance.
- Learning Outcome: I gained experience in designing defensive APIs and ensuring relational integrity in enterprise systems.

6. Debugging and Error Handling

Throughout the internship, I encountered and fixed many errors such as NullReferenceException, bulk insert MAXERRORS, and schema mismatches.

- Technical Challenge: Debugging often consumed entire days, especially when the cause was hidden in another part of the system.
- Reflection: At times it was frustrating, but solving a persistent error gave me a strong sense of accomplishment. I also realized that debugging is one of the most valuable skills in real projects.
- Learning Outcome: I developed systematic debugging techniques, learned how to trace errors using Swagger and SSMS, and improved my resilience when facing unexpected problems.

Reflection

Each task I worked on during the internship gave me a different perspective. The technical challenges (from merging endpoints to debugging migrations) taught me how theoretical knowledge from university is applied in real projects. I realized that even small changes, like user settings, can greatly improve usability, while larger tasks, like enumeration functions, require careful design to ensure efficiency.

On a personal level, I learned the value of patience and persistence. Debugging was often frustrating, but solving issues gave me confidence and showed me the importance of resilience. I also saw how teamwork and senior reviews (Alperen or Hüseyin abi checking code before merging) build accountability and quality.

Overall, the internship helped me bridge academic knowledge with industry practice, develop a stronger coding discipline, and understand that good engineering balances technical accuracy with user trust.

Activity Analysis

Description of the Activity

One of the most critical engineering activities I carried out during my internship was the development of an endpoint in the Indicator Controller that verifies whether an

indicator is already assigned to a Master Product. Before this feature, indicators could be deleted freely, even if they were still in use. This caused serious problems, such as formulas breaking and relational data becoming inconsistent.
My task was to implement a solution that prevented such issues. The endpoint had to:

1. Return the names of the Master Products if the indicator was in use.
2. Return null if the indicator was not used in any Master Product.
3. Block deletion if the indicator was actively linked, while providing a user-friendly warning message on the frontend.

This required coordination between the controller, service, and repository layers of the ASP.NET Core application. I wrote the business logic in the service layer and created optimized queries to fetch the necessary relational data from SQL Server.

Performance Indicators and Metrics

The performance of this new endpoint was evaluated based on:
Data Integrity: Formulas and linked records were now preserved, since indicators could not be deleted if still in use.
System Reliability: Database errors caused by broken foreign keys were eliminated.
Usability: The frontend team was able to integrate clear warnings, preventing users from accidentally deleting critical indicators.

Problematic Issues in the Current Situation

While the feature solved an important problem, it came with its own challenges:
Relational Complexity: Finding the connection path between indicators and Master Products was not straightforward. It required analyzing interface services and identifying common structures used across the system.
Query Design: Writing SQL queries that could handle multiple relational links without slowing down the system was difficult.
Integration: Returning simple, user-friendly error messages (instead of technical database errors) required close collaboration with the frontend team.

Reflection and Learning Outcomes

For me, the most difficult part of this task was discovering the linkage path between indicators and Master Products. Navigating the shared structures in the service interfaces took significant effort, and I realized how important system-level understanding is in backend engineering.

Despite the difficulties, this activity was also one of the most rewarding. It showed me how a relatively small feature could dramatically increase system reliability and user trust. I learned how to design APIs that act as safeguards, how to improve communication between backend and frontend, and most importantly, how to ensure that relational integrity is preserved in enterprise systems.

Project *(only for XX300)*

Description of the Activity

During my internship, I was responsible for implementing User Settings parameters that directly affected how data was processed and validated in the system. These included:

- **Digit Count:** how many digits are shown after the decimal point.
- **Delimiter Type:** how uploaded files (CSV/Excel) are parsed.
- **Sample Size:** how many rows are displayed in "Sample Data."
- **Max Error Threshold:** the maximum allowed number of row-level errors in bulk data insert operations.

At first, I thought this task would be simple and quick to finish. However, understanding the existing backend structure and replacing hard-coded logic with a dynamic system turned out to be much more complex than I expected. What I believed would take half a day initially stretched into three full days of work. Later, with more experience, I realized I could complete similar tasks in less than an hour. This difference showed me how much I had grown in problem-solving efficiency during the internship.

On the frontend side, most of the interface was already prepared. My responsibility was to make the backend flexible so that it could accept and validate dynamic user configurations rather than fixed parameters. This required updates in the controller, improvements in the service layer, and additional validation logic in SQL queries.

Performance Indicators and Metrics

The success of this feature was measured by several factors:

- **Accuracy of Data Parsing:** With delimiter settings, CSV/Excel imports became consistent and prevented column misalignment.
- **System Usability:** Users could preview exactly the number of rows they wanted with the sample size setting.
- **Error Handling Robustness:** The Max Error Threshold allowed imports to fail gracefully instead of crashing completely.
- **User Satisfaction:** Since the frontend team did not need to hard-code configurations anymore, integration was smoother and more flexible.

I personally observed performance overhead when handling large datasets during export operations—errors appeared more slowly when the files were very large. This showed that although the feature worked correctly, optimization was still necessary for scalability.

Problematic Issues in the Current Situation

Despite improvements, some issues remained:
Performance Overhead: Very large files slowed down preview and error detection.
User Misconfiguration: Some users selected unrealistic settings (e.g., max error = 0), which made the system overly strict.
Scalability: Multiple concurrent users with custom configurations increased server load.

Problem Definition

The legitimate Computer Engineering problem identified was:
How to design a user-specific data validation system that maximizes flexibility without degrading performance or reliability?

Proposed Solution and Analysis

To address this, I suggested improvements such as:
Improved Defaults: System-recommended default values (e.g., max error = 100) to avoid unrealistic user settings.

Caching Mechanisms: Caching common preview sizes to prevent repeated heavy queries.
Asynchronous Processing: Handling large previews in background tasks to keep the UI responsive.
Validation Layer: Middleware checks to block invalid user settings before execution.

While I solved the main implementation independently, my mentor Alperen provided hints when I was stuck. This helped me learn without relying too much on others, and finishing the work myself was very satisfying.

Literature Survey and Alternative Approaches

Academic research in data quality management and ETL (Extract–Transform–Load) systems offers several alternative approaches:

- **Adaptive Thresholding (Kandel et al., 2011):** dynamically adjust error tolerance based on dataset size.
- **User-Centric Defaults (Nargesian et al., 2019):** recommend default values derived from prior user behavior.
- **Incremental Data Loading (Abadi et al., 2016):** load datasets in smaller chunks to minimize the effect of errors.

Incorporating such methods could further optimize Lumnion's data validation pipeline and enhance both performance and user experience.

Conclusion

This project showed me how backend engineering and user experience are directly connected. At first, I underestimated the complexity of the task, but working through it taught me how small configuration features can have large impacts on usability and performance. More importantly, I learned how to approach a problem step by step: analyzing requirements, implementing changes, validating with real data, and considering improvements through academic research.

By completing this task, I gained not only technical knowledge in ASP.NET Core, SQL Server, and API design but also practical lessons about efficiency, usability, and independent problem-solving.

3. Conclusions *(maximum one page)*

During my internship at Lumnion Bilişim Teknolojileri, I engaged in a wide range of software engineering activities, including backend development, database management, and system validation. My main contributions were:

- Developing and refactoring backend endpoints such as the Save Rule endpoint.
- Designing enumeration functions in SQL for dynamic actuarial data handling.
- Managing Entity Framework Core migrations and schema updates.
- Implementing user-specific settings (delimiter, digit count, sample size, error thresholds).
- Creating validation endpoints to ensure indicator–master product integrity.
- Debugging and solving runtime issues such as null references and bulk insert errors.

Through these activities, I learned how my theoretical knowledge in database systems, software architecture, and algorithms translates into enterprise-level applications. I also developed practical skills in ASP.NET Core, SQL Server, and Entity Framework Core, while gaining professional habits such as systematic debugging, clean code writing, and effective use of Git branching workflows.

The working environment I experienced was one of the most valuable parts of the internship. Meetings were always held with strong discipline and started on time, yet there was never unnecessary pressure. If someone could not attend, no one reacted negatively; instead, there was an atmosphere of understanding and support. Team members helped each other frequently, and even if someone could not complete a task during the day, they made sure to contribute later. This balance of discipline and flexibility created a highly collaborative and motivating workplace culture.

From a sectoral perspective, the InsurTech industry continues to grow rapidly, with Lumnion holding a strong position domestically. However, competing in international markets presents challenges, particularly against well-established players like Prophet. The major technical challenges of the sector include handling massive datasets, integrating machine learning into production systems, and ensuring compliance with complex financial regulations.

Looking into the next five to ten years, I believe the industry will be significantly shaped by cryptology and cybersecurity. As insurance products increasingly rely on large datasets and cloud-based infrastructures, protecting data integrity and privacy will be as important as developing accurate models. Innovations in cryptology will enhance secure transactions, while advancements in cybersecurity will protect sensitive financial and customer data against evolving threats. These developments will complement ongoing transformations driven by AI, big data, and automation, making InsurTech both more powerful and more responsible.

In conclusion, this internship not only strengthened my technical abilities but also gave me a clear view of the professional responsibilities and future challenges in the InsurTech

sector. I finished the program with stronger coding skills, a deeper appreciation of teamwork, and a sharper vision of how technology and ethics intersect in real-world engineering.

a. Impact *(minimum one page)*

During my internship at Lumnion Bilişim Teknolojileri, I realized that even the technical tasks I contributed to—such as developing backend endpoints, optimizing SQL queries, or implementing user settings—can have a broader global, economic, environmental, and societal impact. While at first these seemed like purely technical details, I gradually saw how they connect to larger issues such as international compliance, financial efficiency, and customer trust.

Global Impact

Insurance is by nature a global industry, and the software solutions developed at Lumnion are designed not only for domestic use but also for international clients. Although I personally did not attend international client meetings, I observed how senior staff such as the CEO and chief analysts frequently engaged with foreign partners.

Economic Impact

The economic effect of software engineering in the InsurTech sector is significant. By improving reliability and preventing errors, we help companies reduce operational costs. For example, the Max Error Threshold feature I worked on prevented entire bulk data uploads from crashing, which saves both time and money. In addition to error reduction, there is also a strong element of labor efficiency. Automating repetitive tasks reduces the need for manual corrections and allows engineers and actuaries to focus on higher-value analytical work. In the long run, this improves company profitability and ensures customers receive fairer and more transparent premium calculations.

Environmental Impact

Although software engineering does not directly produce physical waste, its environmental footprint comes from the computational power used by servers and cloud services. By writing optimized queries and designing more efficient data validation logic, the system consumes fewer resources on cloud platforms such as Azure and AWS. This indirectly contributes to lower energy consumption, which is critical given the increasing energy demands of large-scale data processing. While I did not work directly on cloud resource management, my tasks still played a role in minimizing unnecessary processing, which is a subtle but meaningful contribution to sustainability.

Societal Impact

Perhaps the most visible impact is at the societal level. Insurance companies play a crucial role in protecting individuals and businesses from financial risks. If the software behind these companies is unreliable or prone to errors, the trust of customers can be severely damaged. By implementing safeguards such as indicator, master product validation and user-specific settings, I contributed to ensuring that pricing models remain consistent and transparent. This benefits both sides:

- For customers, it guarantees fairer and more accurate premiums.
- For companies, it builds credibility and reduces disputes over pricing or coverage.

A reliable InsurTech solution, therefore, strengthens the overall relationship between insurers and society by making the system more equitable and trustworthy.

Conclusion

In short, my internship taught me that engineering solutions are never isolated. A single backend improvement can have ripple effects: globally, it supports compliance and competitiveness; economically, it saves costs and labor; environmentally, it reduces energy usage; and socially, it strengthens trust and fairness in financial systems. As an engineering student, this realization was important because it showed me how technical work contributes to broader global, economic, environmental, and societal outcomes, and that even small changes at the code level can create a significant impact.

b. Team Work *(maximum one page)*

In the CashFlow team, I worked closely with both backend and frontend developers, each with clearly defined responsibilities.

On the backend side, our team included:

- **Alperen (Senior Backend Engineer):** He was not only responsible for the core architecture but also took care of company-level setup tasks like deployments and environment configurations. Nothing could be pushed to the backend without his review and approval, which ensured code quality and stability.
- **Arca (Junior Backend Engineer):** He mainly focused on running processes and operational tasks, ensuring that the system was functioning correctly during development.
- **Me (Intern Backend Engineer):** At first, I started with smaller, slower tasks as I was still learning. Over time, I moved on to handling general backend work, such as developing endpoints, debugging issues, and working with migrations. This helped me feel like a true contributor to the backend team.

On the frontend side, the team included:

- **Hüseyin (Senior Frontend Engineer):** He had the final say in frontend development. Nothing could be pushed to production without his approval, which maintained a consistent and reliable UI/UX.
- **Berke (Junior Frontend Engineer):** He supported the development process by implementing features and assisting with frontend–backend integration.
- **Sena (Intern Frontend Engineer):** She contributed by learning through hands-on tasks and supporting the junior and senior developers with smaller assignments.

The dynamic of the team was clear: seniors set the standards and had the final approvals, juniors provided the active coding support, and interns learned while contributing real tasks. I found this structure very effective. It was motivating for me to see my progress—from starting slowly to later taking on regular backend responsibilities, and to observe how collaboration and mentorship ensured smooth teamwork.

c. Self-directed Learning *(minimum one page)*

One of the most important lessons I gained from this internship was the value of self-directed and lifelong learning. From the very first week, I understood that the technologies used in the CashFlow project—such as ASP.NET Core, Entity Framework, SQL Server, Docker, and GitHub workflows—were beyond the level I had fully studied at university. In order to contribute effectively, I realized that I had to take initiative and build my knowledge independently.

At the beginning, I followed the recommendations of the backend and frontend developers and enrolled in Udemy courses on ASP.NET Core and SQL/Entity Framework. These courses helped me strengthen my foundation, but I quickly noticed that the material did not always match the exact problems I was facing in the project. For example, while the course explained the basics of migrations and entity relationships, the real challenge in my tasks was debugging unexpected runtime errors and adapting schema changes to a live project. At first this was discouraging, but later I realized that the value of courses lies in giving a conceptual map, while the real learning comes from hands-on experimentation and problem-solving in real projects.

By the second week, during sprint planning, I was assigned a small task, but my personal goal was to finish another course in parallel. Even though the tasks seemed simple, I soon learned that progress required not just following instructions, but also curiosity and initiative. For example, when implementing backend endpoints, I often stopped and researched not only how to fix an error, but also why it was happening. This habit of “digging deeper” became one of my strongest learning strategies.

I also benefited a lot from mentorship and observation. I frequently consulted Alperen, the senior backend engineer, about deployment processes. He introduced me both to Lumnion’s internal setup (using Docker, Git branching, and application installation) and to how deployments were handled for external companies via virtual machines. These insights showed me how large-scale software systems are managed in real life, going far beyond what I had learned in class. Although Alperen sometimes gave me hints, he encouraged me to solve problems independently. Completing tasks on my own, after struggling for days, gave me confidence and reinforced my learning far more than simply being told the answer.

Another area I explored with Alperen was ethical hacking. While this was more of a side curiosity, we experimented with small exercises and even discussed potential issues related to CashFlow tokens. This made me realize how important security is in financial software, and why engineers must always think about vulnerabilities as well as features. Even though these explorations were not part of my official assignments, they broadened my perspective and motivated me to study security more seriously in the future.

Reflecting on my experience, I would say the most valuable learning method was independent exploration. While online courses gave me a base, they did not immediately solve the problems I faced. Instead, as I experimented with the code, broke things, and traced errors, I started to connect the theoretical knowledge from courses to the actual structures of the project. This way, learning became an active process: not only consuming information but also applying, testing, and reflecting on it.

In conclusion, this internship convinced me that self-learning and continuous improvement are essential in software engineering. Technologies change too quickly for university courses or company training alone to be enough. An engineer must stay curious, take responsibility for their own growth, and build a positive attitude toward lifelong learning. For me, this meant combining online resources with hands-on problem solving, asking questions when necessary, and never being afraid to explore side topics like deployment and security. This mindset will remain central to my professional practice in the future, as I adapt to new technologies such as AI, big data, and cloud computing.

References

1- Microsoft, 2025. ASP.NET Core Documentation.
https://learn.microsoft.com/en-us/aspnet/core/

2- Microsoft, 2025. Entity Framework Core Documentation.
https://learn.microsoft.com/en-us/ef/core/

3- Microsoft, 2025. SQL Server Technical Documentation.
https://learn.microsoft.com/en-us/sql/

4- International Financial Reporting Standards (IFRS), 2025. IFRS 17 Insurance Contracts.
https://www.ifrs.org/issued-standards/list-of-standards/ifrs-17-insurance-contracts/

5- Solvency II Directive, 2025. Directive 2009/138/EC of the European Parliament and of the Council.
https://ec.europa.eu/info/business-economy-euro/banking-and-finance/insurance-and-pensions/risk-management/solvency-2_en

6- IEEE/ACM, 2025. Software Engineering Code of Ethics.
https://ethics.acm.org/code-of-ethics/software-engineering-code/

7- Kandel, S., Paepcke, A., Hellerstein, J. and Heer, J., 2011. Wrangler: Interactive visual specification of data transformation scripts. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*, pp.3363–3372.

8- Nargesian, F., Zhu, E., Miller, R.J., Pu, K.Q. and Zhang, H., 2019. Table Union Search on Open Data. *Proceedings of the VLDB Endowment*, 11(7), pp.813–825.

9- Abadi, D.J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J.H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y. and Zdonik, S., 2016. The Design of the Borealis Stream Processing Engine. *Foundations and Trends in Databases*, 4(1), pp.1–123.

10- Lumnion, 2025. Lumnion Official Website.
https://www.lumnion.com/

Appendix 1: **Daily Activity Tables**

My first day started with an introduction to the company and the CashFlow project. I spent most of the day setting up my development environment and making sure all required tools like **Visual Studio, SQL Server Management Studio, and Git** were running properly. I also joined my first daily meeting at 9:00, where I observed how tasks were shared among the team. In the afternoon, I began a **Udemy course** recommended by my teammates to strengthen my backend knowledge.

Date	Supervisor's Name	Signature
07.07.2025	Eray Alpan	

On the second day, I worked from home and continued with the Udemy course. I also spent time observing how backend developers like Alperen and Arca handled migrations and database operations. Even though I wasn't coding yet, I learned a lot by watching how they debugged problems in the project. This gave me a better picture of the workflow and the challenges I would face later.

Date	Supervisor's Name	Signature
08.07.2025	Eray Alpan	

I attended the daily stand-up in the morning and listened to updates from the frontend and backend sides. I continued with the Udemy course, focusing on **Entity Framework basics**. Later, there was an investor meeting where I joined as a listener. Even though I didn't contribute, it was interesting to see how technical progress is reported to stakeholders.

Date	Supervisor's Name	Signature
09.07.2025	Eray Alpan	

This was my second day at the office. I joined the **product meeting** about the company's other platform. Most of the day I continued my course and asked Alperen a few questions about system setup and deployments. I also watched closely how the frontend team worked with API responses, which gave me ideas about how backend and frontend interact.

Date	Supervisor's Name	Signature
10.07.2025	Eray Alpan	

I spent the day at home, mainly focused on finishing sections of my Udemy course. I also shadowed some of Arca's work related to running services. This gave me more understanding of how backend jobs are triggered and monitored. It was a quiet but useful day for learning.

Date	Supervisor's Name	Signature
11.07.2025	Eray Alpan	

This week started with my first **sprint planning meeting**, where the next two weeks of tasks were discussed. I still wasn't assigned heavy work, but I was given a **small task related to migrations**. The task was to review how new columns were being added and to check error messages in SQL Server. Alongside this, I continued another Udemy course to strengthen my backend fundamentals.

Date	Supervisor's Name	Signature
14.07.2025	Eray Alpan	

I joined the daily in the morning, then spent most of the day finalizing the second Udemy course. I also did a **small fix on nullable IDs** with the guidance of Arca. It was the first time I touched actual project code, even though it was something minor. This small success motivated me a lot.

Date	Supervisor's Name	Signature
16.07.2025	Eray Alpan	

On Thursday, I went to the office again. After the daily, I joined the **product meeting**. I also handled another small backend task that involved checking for errors in database queries. Even though the task was not complex, it was interesting to see how careful validation prevents bigger errors later.

Date	Supervisor's Name	Signature
17.07.2025	Eray Alpan	

From home, I reviewed the design of the **Save Rule endpoint**. My job was not to code it yet, but to understand how the existing Create and Update endpoints worked. I took notes about their similarities and differences. This preparation was important because from the next week onwards, I would start working on larger backend tasks

Date	Supervisor's Name	Signature
18.07.2025	Eray Alpan	

The week started with a **sprint planning session**. In this sprint, my main focus was the **Save Rule endpoint**. At the office, Alperen explained how Create and Update were currently separated and why merging them into Save would make the system cleaner. I studied the controller and service layers, identifying repeated logic that could be consolidated. This gave me a much clearer roadmap for the coming days.

Date	Supervisor's Name	Signature
21.07.2025	Eray Alpan	

I started implementing the Save Rule endpoint at home. My work began with the **controller layer**, adding the new route and defining request parameters. It was challenging because I had to ensure the endpoint could handle both new and existing IDs. I also reviewed Entity Framework's transaction handling to prevent data corruption if something went wrong.

Date	Supervisor's Name	Signature
22.07.2025	Eray Alpan	

The coding continued. I worked mainly on the **service layer**, where I had to separate logic for create vs. update without duplicating code. It took me a while to debug null reference issues, but with Arca's help I managed to resolve them. In the afternoon, there was an investor meeting. Although I mostly listened, it helped me understand how backend stability directly affects customer confidence.

Date	Supervisor's Name	Signature
23.07.2025	Eray Alpan	

At the office, I tested the Save Rule endpoint using **Swagger and Postman**. The feature worked in most cases, but I found a few bugs when handling missing IDs. Alperen suggested adding clearer validation messages so the frontend team could easily display them. Later, I joined the weekly **product meeting** about the platform product. It was useful to see how different teams coordinate.

Date	Supervisor's Name	Signature
24.07.2025	Eray Alpan	

I finalized the Save Rule endpoint and documented my changes. This included writing down the request/response formats and validation rules. I also cleaned up some redundant code from the old endpoints. This task was the first real feature I completed almost independently, which gave me a big confidence boost.

Date	Supervisor's Name	Signature
25.07.2025	Eray Alpan	

A new sprint started, and my task was to begin work on **enumeration functions in SQL**. I started with INT_TO_ENUM and ENUM_SIZE. Alperen explained how actuarial models often depend on these mappings, so the functions had to be reliable. I created draft SQL functions and tested them with simple cases in SSMS.

Date	Supervisor's Name	Signature
28.07.2025	Eray Alpan	

I refined INT_TO_ENUM by making sure it could handle invalid inputs gracefully (returning null). Then I worked on ENUM_SIZE, which required counting the number of key–value pairs in a table. It looked simple at first, but I had to ensure it worked across different enumeration tables dynamically. Debugging this taught me a lot about SQL function design.

Date	Supervisor's Name	Signature
29.07.2025	Eray Alpan	

I started exploring the more complex **READ_FROM_TABLE** function. This required retrieving a value by matching multiple row headers. The logic was tricky because the number of headers could vary. I spent most of the day analyzing how to dynamically construct SQL queries to handle different cases.

Date	Supervisor's Name	Signature
30.07.2025	Eray Alpan	

I continued implementing READ_FROM_TABLE. At the office, I asked Alperen for help with handling text vs. numeric types. Together, we tested queries to ensure correct results when row headers were strings. Later, in the **product meeting**, I realized how important these functions were because they directly supported actuarial formulas that customers rely on.

Date	Supervisor's Name	Signature
31.07.2025	Eray Alpan	

I completed the first working version of READ_FROM_TABLE and tested it with sample tables. Some cases worked perfectly, but others failed when row headers were missing. I documented the problems and planned fixes for the next sprint. Even though it wasn't perfect, I felt satisfied with making real progress on such a complex function.

Date	Supervisor's Name	Signature
01.08.2025	Eray Alpan	

A new sprint started, and my focus shifted to **User Settings integration**. I began with the **delimiter** and **digit count** parameters. At the office, I reviewed how these settings were stored in the database and how they should be applied dynamically to user operations. It was interesting to see how small configuration changes could affect the whole data input pipeline.

Date	Supervisor's Name	Signature
04.08.2025	Eray Alpan	

I continued with User Settings, this time working on **sample size**. The challenge was to ensure the system could display a user-defined number of rows in the preview without causing performance issues. I tested with different sample sizes and realized that very large values slowed down the system, so we added sensible limits.

Date	Supervisor's Name	Signature
05.08.2025	Eray Alpan	

My task for the day was to connect the **max error threshold** parameter to bulk insert operations. This meant that if the user set a threshold of, for example, 50, the import would stop only after 50 errors instead of failing immediately. I modified the controller to pass this parameter to the service layer. It was a challenging but very practical task since it improved usability for real customers.

Date	Supervisor's Name	Signature
06.08.2025	Eray Alpan	

At the office, I tested all of the User Settings together: delimiter, digit count, sample size, and max error threshold. There were some issues with how the frontend displayed decimals, so I worked with Hüseyin to make sure the backend responses were aligned with the UI. In the afternoon, we had the **weekly product meeting**, where I presented the progress on these settings.

Date	Supervisor's Name	Signature
07.08.2025	Eray Alpan	

I spent most of the day debugging small issues related to User Settings. For example, when the delimiter was not specified, the system sometimes defaulted incorrectly. Fixing these edge cases helped me understand the importance of validation and default values. By the end of the week, the User Settings feature was functioning well.

Date	Supervisor's Name	Signature
08.08.2025	Eray Alpan	

A new sprint began, and I moved on to the **Master Product–Indicator check endpoint**. The goal was to prevent indicators from being deleted if they were still assigned to a Master Product. At the office, I worked with Alperen to design the controller route and service logic. It was exciting because this task required thinking about **data integrity** and relational constraints.

Date	Supervisor's Name	Signature
11.08.2025	Eray Alpan	

I continued coding the endpoint at home. The main challenge was writing a query that could fetch all Master Product names linked to a given indicator. I also made sure the endpoint would return null if no link existed. It was tricky to handle these conditions correctly, but I learned a lot about working with relationships in Entity Framework.

Date	Supervisor's Name	Signature
13.08.2025	Eray Alpan	

I tested the new endpoint thoroughly. When an indicator was linked, the endpoint successfully returned the Master Product names. When not linked, it returned null as expected. During the investor meeting, I realized how important this feature was: preventing broken references builds customer trust in the system. I documented all tests and bug fixes.

Date	Supervisor's Name	Signature
12.08.2025	Eray Alpan	

This was my last full working day at the office. I finalized the Master Product–Indicator endpoint and presented it in the **product meeting**. I also had a closing discussion with the team about what I learned during my internship. Looking back, I was proud of my progress—from starting slowly with courses and small fixes to completing full backend features.

Date	Supervisor's Name	Signature
14.08.2025	Eray Alpan	

Although this was officially my last internship day, I mostly focused on wrapping up documentation and preparing my internship report. I also joined the daily meeting one last time and thanked the team for their support. It was a nice closing to an internship where I learned not just technical skills but also how to work effectively in a real development team.

Date	Supervisor's Name	Signature
15.08.2025	Eray Alpan	

Appendix 2:

```
2 references
public async Task ReloadProductCounts(int productId, IConfiguration configuration)
{
    var product = await _dbContext.Products
        .Include(p => p.SubProducts)
        .FirstOrDefaultAsync(p => p.Id == productId);

    if (product == null)
        throw new Exception($"Product {productId} not found");

    string sourceTable = _dbContext.ModelDatas
        .Where(md => md.Id == product.ModelDataId)
        .Select(md => md.DataTableName)
        .First();

    string connStr = configuration.GetConnectionString("DefaultConnection");
    using var conn = new SqlConnection(connStr);
    await conn.OpenAsync();

    var allGroups = new List<string>();
    string distinctSql = $"SELECT DISTINCT [{product.ProductColumn}] FROM [{sourceTable}]";
    using (var cmd = new SqlCommand(distinctSql, conn))
    using (var reader = await cmd.ExecuteReaderAsync())
    {
        while (await reader.ReadAsync())
        {
            if (reader[0] != DBNull.Value)
                allGroups.Add(reader[0].ToString());
        }
    }
```

Figure 1. Service of ReloaProductCounts function which I coded, PART I.

```
    foreach (var sub in product.SubProducts)
    {
        string sql = $"SELECT COUNT(*) FROM [{sourceTable}] WHERE [{product.ProductColumn}] = @val";
        using var cmd = new SqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@val", sub.Name);

        int count = (int)await cmd.ExecuteScalarAsync();

        sub.Count = count;
    }

    foreach (var group in allGroups)
    {
        if (!product.SubProducts.Any(sp => sp.Name == group))
        {
            string sql = $"SELECT COUNT(*) FROM [{sourceTable}] WHERE [{product.ProductColumn}] = @val";
            using var cmd = new SqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@val", group);

            int count = (int)await cmd.ExecuteScalarAsync();

            product.SubProducts.Add(new SubProduct
            {
                Name = group,
                DataTableName = $"{sourceTable}_{group}",
                Count = count
            });
        }
    }

    await _dbContext.SaveChangesAsync();

    product.SubProducts = product.SubProducts
        .Where(sp => sp.Count > 0)
        .ToList();
}
}
```

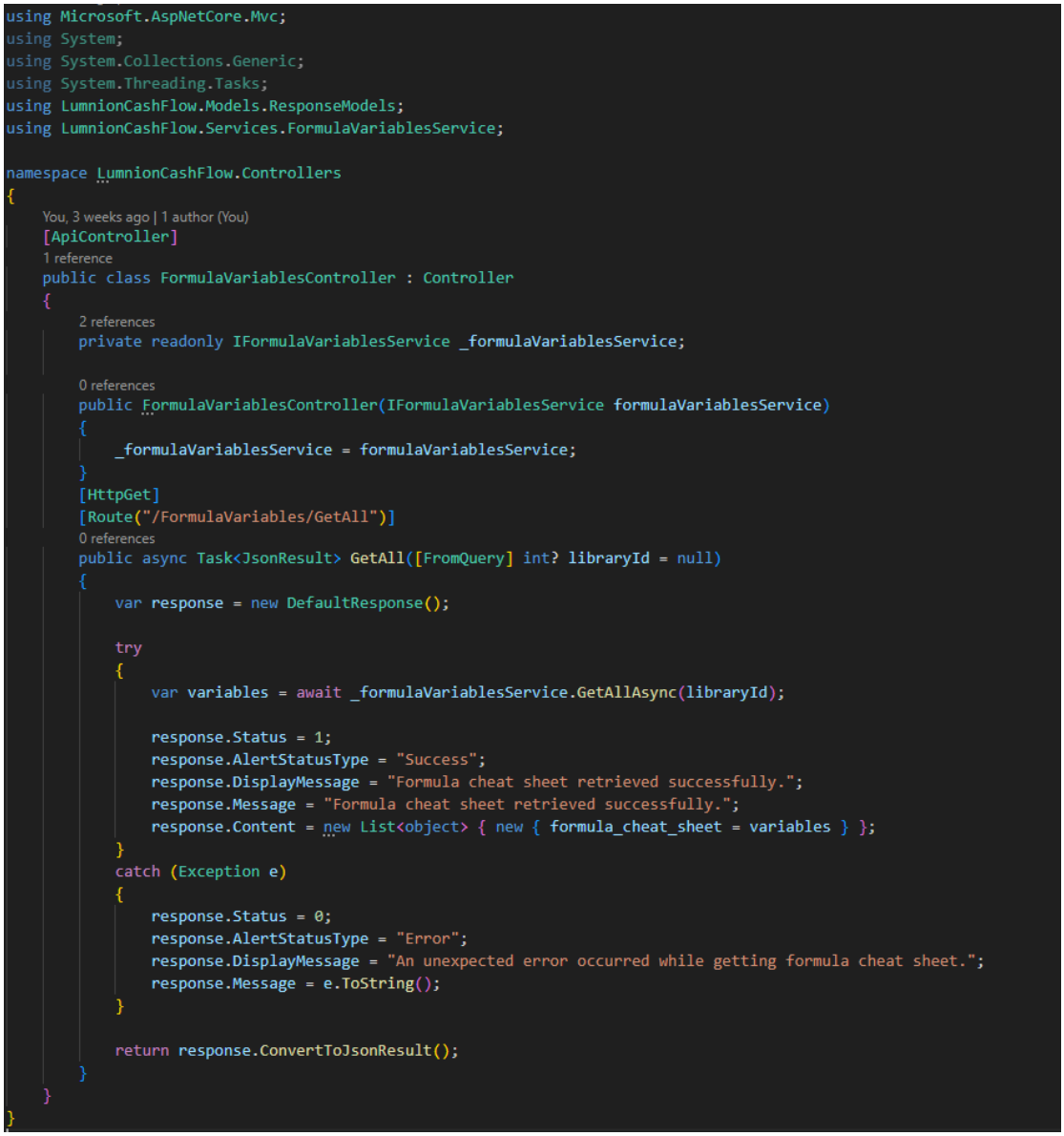
Figure 2 Service of ReloaProductCounts function which I coded, PART II.

```
[HttpPost]
[Route("/Product/ReloadCounts")]
0 references
public async Task<JsonResult> ReloadCounts([FromQuery] int productId)
{
    var response = new DefaultResponse();
    try
    {
        await _productService.ReloadProductCounts(productId, _configuration);
        response.Status = 1;
        response.AlertStatusType = "Success";
        response.DisplayMessage = "Product counts reloaded successfully.";
        response.Message = "Product counts reloaded successfully.";
    }
    catch (Exception ex)
    {
        response.Status = 0;
        response.AlertStatusType = "Error";
        response.DisplayMessage = "An error occurred while reloading product counts.";
        response.Message = ex.ToString();
    }
    return response.ConvertToJsonResult();
}
```

Figure 3. EndPoint of ReloaProductCounts function.

```
You, 4 weeks ago | 1 author (You)
1  using LumnionCashFlow.Entities;
2  using LumnionCashFlow.Models;
3
4  namespace LumnionCashFlow.Services.FormulaVariablesService
5  {
       4 references | You, 4 weeks ago | 1 author (You)
6      public interface IFormulaVariablesService
7      {
           2 references
8          Task<IReadOnlyList<object>> GetAllAsync(int? libraryId = null);
9      }
10 }
11
```

Figure 4. Interface Service of FormulaVariableService function which I coded fully.



```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using LumnionCashFlow.Models.ResponseModels;
using LumnionCashFlow.Services.FormulaVariablesService;

namespace LumnionCashFlow.Controllers
{
    You, 3 weeks ago | 1 author (You)
    [ApiController]
    1 reference
    public class FormulaVariablesController : Controller
    {
        2 references
        private readonly IFormulaVariablesService _formulaVariablesService;

        0 references
        public FormulaVariablesController(IFormulaVariablesService formulaVariablesService)
        {
            _formulaVariablesService = formulaVariablesService;
        }
        [HttpGet]
        [Route("/FormulaVariables/GetAll")]
        0 references
        public async Task<JsonResult> GetAll([FromQuery] int? libraryId = null)
        {
            var response = new DefaultResponse();

            try
            {
                var variables = await _formulaVariablesService.GetAllAsync(libraryId);

                response.Status = 1;
                response.AlertStatusType = "Success";
                response.DisplayMessage = "Formula cheat sheet retrieved successfully.";
                response.Message = "Formula cheat sheet retrieved successfully.";
                response.Content = new List<object> { new { formula_cheat_sheet = variables } };
            }
            catch (Exception e)
            {
                response.Status = 0;
                response.AlertStatusType = "Error";
                response.DisplayMessage = "An unexpected error occurred while getting formula cheat sheet.";
                response.Message = e.ToString();
            }

            return response.ConvertToJsonResult();
        }
    }
}
```

Figure 5. Controller of FormulaVariableService function which I coded fully.

```
You, 2 weeks ago | 1 author (You)
using LumnionCashFlow.Entities;
using LumnionCashFlow.Models;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LumnionCashFlow.Services.FormulaVariablesService
{
    2 references | You, 2 weeks ago | 1 author (You)
    public class FormulaVariablesService : IFormulaVariablesService
    {
        5 references
        private readonly ApplicationDbContext _context;    // yeni
        4 references
        private readonly ApplicationDbContext _dbContext;  // eski kullananlar için

        0 references
        public FormulaVariablesService(ApplicationDbContext context)
        {
            _context = context ?? throw new ArgumentNullException(nameof(context));
            _dbContext = _context; // aynı instance
        }

        2 references
        public async Task<IReadOnlyList<object>> GetAllAsync(int? libraryId = null)
        {
            if (libraryId.HasValue)
            {
                if (libraryId.Value <= 0 ||
                    !await _dbContext.Libraries.AnyAsync(l => l.Id == libraryId.Value))
                {
                    throw new ArgumentOutOfRangeException(
                        nameof(libraryId),
                        $"The library with id {libraryId.Value} can't be found."
                    );
                }
            }

            var items = new List<object>();
            // 1) CalculationVariables (dinamik)
            var cvQuery = _dbContext.CalculationVariables
                                    .Include(cv => cv.Formulas)
                                    .AsQueryable();

            if (libraryId.HasValue)
                cvQuery = cvQuery.Where(cv => cv.LibraryId == libraryId.Value);

            var calculationVariables = await cvQuery.AsNoTracking().ToListAsync();

            items.AddRange(
                calculationVariables.Select(cv => (object)new
                {
                    id = cv.Id,
                    title = cv.Name,                          // cv_name
                    type = "Calculation Variable",
                    type_description = (string?)null,
                    description = cv.Description,             // Gross premium at time t
                    example_formula = $"{cv.Name}(t)",         // CV_NAME(t)
                    table_name = "",
                })
            );
```

Figure 6. Service of FormulaVariableService function which I coded fully PART I.

```
    // 2) Generic Table Data (dinamik)
    var genericTables = await _dbContext.GenericTableDatas
                                        .AsNoTracking()
                                        .ToListAsync();

    foreach (var td in genericTables)
    {
        // 3a) ROW HEADER kolonları (LabelNames) → TableData Column (string)
        if (td.LabelNames != null)
        {
            foreach (var rowHeader in td.LabelNames)
            {
                items.Add(new
                {       You, 3 weeks ago • FormulaVariables changes and updates …
                    id = td.Id,
                    title = rowHeader,   // Tabledata_RowHeader_Name
                    type = "Tables' Row Headers",
                    type_description = "string",        // row header tipi string
                    description = $"comes from {td.Name}",
                    table_name = td.Name,
                    example_formula = rowHeader
                });
            }
        }

        // 3b) DEĞER kolonları (DataTableColumns: ad -> sql tip)
        if (td.DataTableColumns != null)
        {
            foreach (var col in td.DataTableColumns)   // Key=column name, Value=sql type
            {
                items.Add(new
                {
                    id = td.Id,
                    title = col.Key,     // Tabledata_Column_Name
                    type = "Tables' Columns",
                    type_description = col.Value,       // Column_Type
                    description = $"comes from {td.Name}",
                    table_name = td.Name,
                    example_formula = col.Key
                });
            }
        }
    }

    // Değer kolonları
    items.AddRange(
    genericTables.Select(td => (object)new
    {
        id = td.Id,
        title = td.Name,                        // Tabledata_Name
        type = "Tables",
        type_description = (string?)null,
        description = "Table Data",
        table_name = td.Name,
        example_formula = td.Name
    })
);
```

Figure 7. Service of FormulaVariableService function which I coded fully PART II.

```
// 3) InputVariables (dinamik)
var inputVariables = libraryId.HasValue
    ? await _context.InputVariables
                    .Where(iv => iv.LibraryId == libraryId.Value)
                    .Include(iv => iv.Product) // Product'ı dahil et
                    .AsNoTracking()
                    .ToListAsync()
    : await _context.InputVariables
                    .Include(iv => iv.Product)
                    .AsNoTracking()
                    .ToListAsync();

var modelDataMap = await _context.ModelDatas
    .ToDictionaryAsync(md => md.Id, md => md.ModelDataName);

items.AddRange(       You, 4 weeks ago • Code helper pard changed …
    inputVariables.Select(iv => (object)new
    {
        id = iv.Id,
        title = iv.ColName,
        type = "Input Variable",
        type_description = iv.ColType,
        description = iv.Product != null &&
                      modelDataMap.TryGetValue(iv.Product.ModelDataId, out var mdName)
            ? $"comes from {mdName}"
            : "comes from unknown modeldata",
        example_formula = iv.ColName,
        table_name = "",
    })
);
```

Figure 8. Service of FormulaVariableService function which I coded fully PART III.

```
//4) Logical & Conditional Expressions
items.AddRange(new object[]
{
    new {
        id = (int?)null,
        title = "OR",
        type = "Logical Expression",
        type_description = (string?)null,
        description = "Logical OR",
        example_formula = " (A) OR (B) ",
        table_name = "",
    },
    new {
        id = (int?)null,
        title = "AND",
        type = "Logical Expression",
        type_description = (string?)null,
        description = "Logical AND",
        example_formula = " (A) AND (B) ",
        table_name = "",
    },      You, 3 weeks ago • FormulaVariables changes and updates
    new {
        id = (int?)null,
        title = "NOT",
        type = "Logical Expression",
        type_description = (string?)null,
        description = "Logical NOT",
        example_formula = " NOT (A) ",
        table_name = "",
    },
    new {
        id = (int?)null,
        title = "NAND",
        type = "Logical Expression",
        type_description = (string?)null,
        description = "NOT(AND)",
        example_formula = " NOT( (A) AND (B) ) ",
        table_name = "",
    },
    new {
        id = (int?)null,
        title = "NOR",
        type = "Logical Expression",
        type_description = (string?)null,
        description = "NOT(OR)",
        example_formula = " NOT( (A) OR (B) ) ",
        table_name = "",
    },
```

Figure 9. Service of FormulaVariableService function which I coded fully PART IV.

```
                new {
                    id = (int?)null,
                    title = "IF / ELSE IF / ELSE",
                    type = "Conditional Expression",
                    type_description = (string?)null,
                    description = "Conditional branching",
                    table_name = "",
                    example_formula =
                    @"/* Example:
IF (a > b) THEN 1
ELSE IF (a = b) THEN 0
ELSE -1
*/
IF (condition1) THEN expr1
ELSE IF (condition2) THEN expr2
ELSE expr3"
                }
            });

            items.AddRange(new object[]
            {
                new {
                    id = (int?)null,
                    title = "INT TO ENUM",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Returns the enum key (text) that corresponds to a numeric value.",
                    table_name = "",
                    example_formula =
                    @"/* Example: INT_TO_ENUM('Alphabet', 2) → 'B' */
INT_TO_ENUM('TABLE_NAME', NUMERIC)"
                },
                new {
                    id = (int?)null,
                    title = "ENUM TO INT",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Converts an enum key (text) to its numeric value.",
                    table_name = "",
                    example_formula =
                    @"/* Example: ENUM_TO_INT('Alphabet','B') → 2 */
ENUM_TO_INT('TABLE_NAME','KEY')"
                },
                new {
                    id = (int?)null,
                    title = "ENUM TO TEXT",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Returns the enum's textual key when given either a numeric value or a key.",
                    table_name = "",
                    example_formula =
                    @"/* Examples:
ENUM_TO_TEXT('Alphabet', 3) → 'C'
ENUM_TO_TEXT('Alphabet', 'C') → 'C' */
ENUM_TO_TEXT('TABLE_NAME', ARG)"
                },
```

Figure 10. Service of FormulaVariableService function which I coded fully PART V.

```
                new {
                    id = (int?)null,
                    title = "TEXT TO ENUM",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Converts a textual key into its corresponding numeric value.",
                    table_name = "",
                    example_formula =
                    @"/* Example: TEXT_TO_ENUM('Alphabet', 'C') → 3 */
TEXT_TO_ENUM('TABLE_NAME', 'TEXT')"
                },
                new {
                    id = (int?)null,
                    title = "ENUM_SIZE",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Returns the total number of elements in the enum table.",
                    table_name = "",
                    example_formula =
                    @"/* Example: ENUM_SIZE('Alphabet') → 26 */
ENUM_SIZE('TABLE_NAME')"
                },
                new {
                    id = (int?)null,
                    title = "READ_FROM_TABLE",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Gets a column value from a table by matching given row header values.",
                    table_name = "",
                    example_formula =
                    @"/* Example: READ_FROM_TABLE('ExchangeRates', ['USD','TRY'], 'Rate', FLOAT) */
READ_FROM_TABLE('TABLE_NAME', [ROW_HEADER_1, ROW_HEADER_2, ...], 'VALUE_COLUMN', (optional) TYPE)"
                },
                new {
                    id = (int?)null,
                    title = "MIN",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Returns the smaller of two expressions.",
                    table_name = "",
                    example_formula =
                    @"/* Example: MIN(a, b) → CASE WHEN a < b THEN a ELSE b END */
MIN(expr1, expr2)"
                },
                new {
                    id = (int?)null,
                    title = "MAX",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Returns the larger of two expressions.",
                    table_name = "",
                    example_formula =
                    @"/* Example: MAX(a, b) → CASE WHEN a > b THEN a ELSE b END */
MAX(expr1, expr2)"
                },
                new {
                    id = (int?)null,
                    title = "DECLARE",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Declares a scalar variable, optionally with type and/or an initial value.",
                    table_name = "",
                    example_formula =
                    @"/* Examples: DECLARE x AS Number;  |  DECLARE x := 5;  |  DECLARE x AS Number := 5; */
DECLARE MyVar AS Number"
```

Figure 11. Service of FormulaVariableService function which I coded fully PART VI.

```
                new {
                    id = (int?)null,
                    title = "FOR EACH",
                    type = "Loop",
                    type_description = (string?)null,
                    description = "Iterates from a start value to an end value.",
                    table_name = "",
                    example_formula =
                    @"/* Example:
FOR EACH i = 1 TO 3
/* body */
NEXT
*/
FOR EACH i = 1 TO 10
/* body */
NEXT"
                },
                new {
                    id = (int?)null,
                    title = "PUBLIC",
                    type = "Declaration",
                    type_description = (string?)null,
                    description = "Declares a public scalar variable with a type.",
                    table_name = "",
                    example_formula =
                    @"/* Example: PUBLIC Threshold AS Number */
PUBLIC Threshold AS Number"
                },
                new {
                    id = (int?)null,
                    title = "PRIVATE",
                    type = "Declaration",
                    type_description = (string?)null,
                    description = "Declares a private scalar variable with a type.",
                    table_name = "",
                    example_formula =
                    @"/* Example: PRIVATE TempValue AS Number */
PRIVATE TempValue AS Number"
                    },
                    new {
                        id = (int?)null,
                        title = "ENUM / END ENUM",
                        type = "Declaration",
                        type_description = (string?)null,
                        description = "Defines an enumeration block where each member is assigned an incremental integer starting from 1.",
                        table_name = "",
                        example_formula =
                        @"/* Example:
ENUM Colors
    Red
    Green
    Blue
END ENUM*/

ENUM EnumName
    Member1
    Member2
    Member3
END ENUM"
```

Figure 12. Service of FormulaVariableService function which I coded fully PART VII.

```
                    },
                    new {
                    id = (int?)null,
                    title = "MULT",
                    type = "Built-in Function",
                    type_description = (string?)null,
                    description = "Checks if the first number can be divided evenly by the second number.",
                    table_name = "",
                    example_formula =
                    @"/* Example: MULT(10, 2) → 1
Example: MULT(10, 3) → 0 */
MULT(NUMERATOR, DENOMINATOR)"
    }
            });
            return items;
        }
```

Figure 13. Service of FormulaVariableService function which I coded fully PART VIII.

Internship Documents
CHECKLIST
Department Name **Engineering Practice**
MEF University

For the completeness of your internship report submission, you are held responsible to check the submission of the following items. The marked final checklist should be included to your submitted internship report as the last page. Student Evaluation Survey has to be submitted only through SIS.

- ☐ Internship Application and Acceptance Form signed/stamped? (Critical)
- ☐ Internship Report (Critical)
- ☐ Internship Evaluation Form enclosed/sealed (Critical)

  (needs to be filled with pen, or via the link sent directly to the company. Either way, form should be signed and enclosed)
- ☐ Daily activity pages signed? (Critical)
- ☐ Additional material (such as source codes) attached to the submitted report (Non-Critical)
- ☐ Internship Report bound and plastic-covered? (Critical)
- ☐ Internship Report and all accompanying material submitted in an envelope.
- ☐ Hereby, I accept liability for the accuracy and integrity of the submitted contents.

Student Name:

Signature:

Date: