

System Description: Small Loan Application System

Team: Alejandro Ballesteros Perez, Phasha Davrishev, Roman Krutsko, Khamidjon Khamidov

Task 1: Description

The Small Loan Application System facilitates customers in obtaining small loans ranging from 500 to 15,000 EUR when purchasing products from shops affiliated with an organization. The system involves multiple user roles, including **Customers** and **Employees** (Organization representatives), to process and approve loan applications efficiently.

Classes, Attributes, and Relationships

Class	Attributes	Relationships
Customer	<ul style="list-style-type: none">- CustomerID- Identity Code- Name- Contact Information- Gender- Birthday- Citizenship- Occupation	<ul style="list-style-type: none">- Can have multiple Loan Applications (1..n)
Employee	<ul style="list-style-type: none">- EmployeeID- Experience- Position- Shift	<ul style="list-style-type: none">- Approves Loan Applications (1..n)- Associated with one Organization (1..1)
Loan Application	<ul style="list-style-type: none">- ApplicationID- LoanAmount- LoanPeriod- Interest Rate- Decision Status (positive/negative)- Creation Date- Product- Customer- CustomerSignature- EmployeeSignature	<ul style="list-style-type: none">- Linked to one Customer (1..1)- Linked to one Product (1..1)- Processed at one Shop (1..1)- Approved by one Employee (1..1)- May result in one Contract (1..1)
Contract	<ul style="list-style-type: none">- ContractID- Status- TerminationDate- Application- Duration	<ul style="list-style-type: none">- Originates from one Loan Application (1..1)

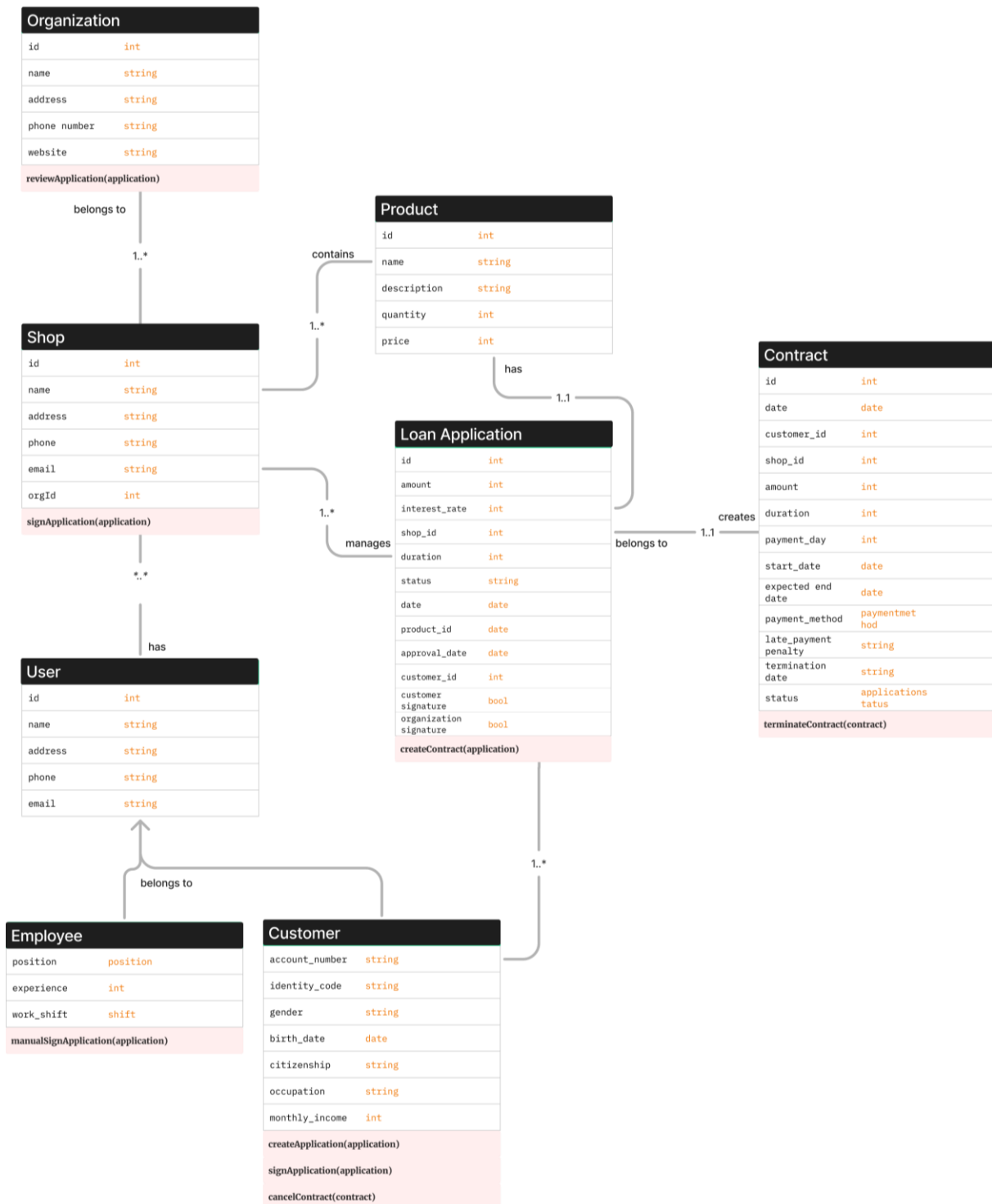
	- Amount - Shop - Start Date	
Organization	- OrganizationID - Name - Address - Contact Info	- Owns multiple Shops (1..n) - Employs multiple Employees (1..n)
Shop	- ShopID - Address - ContactInfo	- Belongs to one Organization (1..1) - Offers multiple Products (1..n) - Facilitates multiple Loan Applications (1..n)
Product	- ProductID - Name - Price - Description - Quantity	- Available at multiple Shops (1..n) - Linked to multiple Loan Applications (1..n)

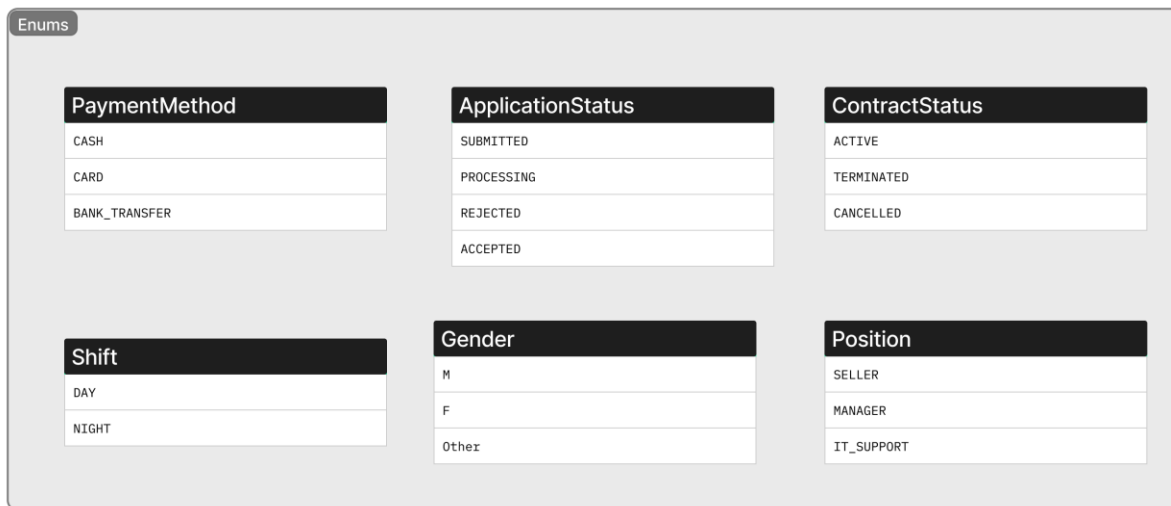
System Workflow:

- Customer visits a Shop and selects a Product.
- If interested in financing, a Loan Application is created, incorporating data from the Customer, Product, Shop, and Organization, along with loan amount and period.
- The Loan Application undergoes an automatic decision process by the Organization, resulting in a Decision Status of positive or negative.
- Positive: The application can proceed to become a Contract.
- Negative: The application is blocked from further progression. Its possible to have manual review by Employee
- For a positive decision, both the Customer and an Employee must approve (sign) the Loan Application.
- Once approved by both parties, the Loan Application transitions into a Contract.

Task 2: Class Model

Link to model: <https://www.figma.com/board/pSn5QPRGnAveXV0f3bPlea/Loan-Application?node-id=0-1&t=xZWbkpfqZ6srQtRz-1>





Task 3: Class Model

Our first choice for the current task of creating a class model was Figma. It is a cloud-based design tool mostly used for UI/UX design, but in our case, it also suits nicely to effectively design class diagrams due to its flexibility. Its interface and familiarity with our team made this tool a relatively easy choice for creating a simple class diagram. Since there are literally no restrictions to designing things in it, it is easy to visually represent class structures and relationships with Figma's extensive library of shapes, and connectors. As it is in the cloud, our team can work simultaneously on one project or diagram in an online workspace, which is a great advantage of using this tool. The ability to export designs in various formats also ensures good integration with other tools during the development process.

Our choice for future diagram modelling will be Visual Paradigm, which is as they say "the market #1 visual modeling and diagramming platform" designed to help create various types of diagrams. An important moment for our team in terms of choosing a tool for this matter was the presence of a free trial or community version in the case of Visual Paradigm, which makes it suitable for all kinds of users, especially students. This tool is also an online platform, which provides flexibility in working on projects, if you connect to Visual Paradigm Online, it reveals a lot of new team tools to conveniently manage the work. Your projects and diagrams can be saved to an online workspace. Diagrams specifically can be edited from both online and desktop versions. Additionally, the tool includes a version control system, allowing users to track changes, revert to previous versions, and manage the progress of the diagram. Users can upload work through commits (like in GitHub), however, in Community Edition only 5 commits per day are allowed.





































Task 3: (Additional Point)

Regarding Figma, unfortunately, it doesn't provide the ability to generate code from a class diagram, since it is not specifically a tool to design these structures, but rather generally for UI/UX design. As for Visual Paradigm, has a very nice feature of generating not only application code from diagrams but also the code for databases. It supports various programming languages, such as Java and C#. This tool has great support for Java persistent code, including static methods, factory classes, Data Access Objects (DAO), Plain Old Java Objects (POJO), or even just the mapping between objects and database entities. However, we have not yet tried this feature of Visual Paradigm, since application class mode is yet to be created.

Task 4: Evidence of Teamwork

Link to Gitlab Repository: <https://gitlab.cs.ut.ee/alejandro1/systems-modelling>

Screenshot of commit history:

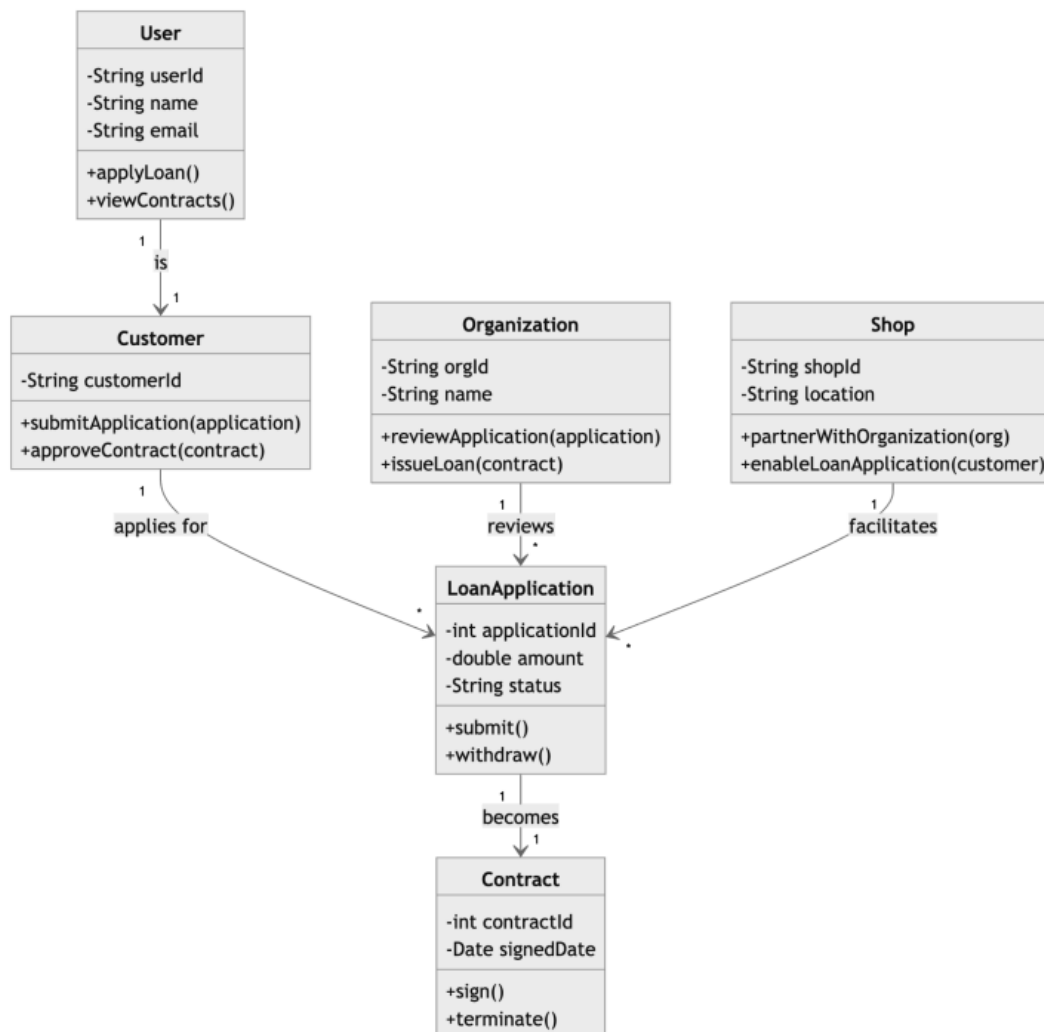
Oct 15, 2024		
	update class diagram khamidjon authored 2 hours ago	1e662aa6  
	Merge branch 'main' of gitlab.cs.ut.ee:alejandro1/systems-modelling PhashaDavrishev authored 4 hours ago	e827f52c  
	update a bit PhashaDavrishev authored 4 hours ago	cdf07d92  
	Task 3, small fix for pdf document romankrutsko authored 7 hours ago	cc3dfed0  
Oct 14, 2024		
	Task 3 romankrutsko authored 19 hours ago	7fb50793  
Oct 13, 2024		
	Replace SystemsModellingTask1.pdf alejandro1 authored 1 day ago	c3b96fd8  
	Replace SystemsModellingTask1.pdf alejandro1 authored 1 day ago	5e6ee8f7  
	task 5 PhashaDavrishev authored 1 day ago	1b3feb4c  
Sep 24, 2024		
	Class diagram for loan application khamidjon.khamidov authored 3 weeks ago	58c89ab2  
	HW1 Upload Task 1 Alejandro Ballesteros Perez authored 3 weeks ago	f5592dfe  
	Add dir HW1 Alejandro Ballesteros Perez authored 3 weeks ago	9a32b415  
	Initial commit Alejandro Ballesteros Perez authored 3 weeks ago	33808dfd  

Task 5: Explore and document Generative AI capabilities

The application we have used for generating the UML class diagram is <https://diagrammingai.com>.

It is a GPT-based GenAI tool, which can create various types of diagrams, including but not limited to flowcharts, sequence, entity relationship, class diagrams, given a prompt or an image. We have chosen DiagrammingAI, for its specialized capabilities in creating the diagrams, and its ability to iteratively refine diagrams through prompts. The prompt used for generating the diagram is as follows

- The system facilitates a small loan process, allowing users to apply for loans between 500 to 15,000 EUR, sign contracts, and receive funds in their bank accounts. Loan decisions are nearly instantaneous for most applicants. It manages various user types, including customers and organization representatives, to streamline loan processing. Key entities include User, Customer, Loan Application, Contract, Organization (loan provider), and Shop (partner store for loan applications). Customers can apply for multiple loans through this system. A loan application signed both by the Organization and the customer becomes a Contract. Below is the output:



Considering that the GenAI tool was provided with the limited description, the result can be counted as satisfactory. Although the number of attributes is insufficient (which can be justified by the lack of context), the ones present were aligned with the diagram we created. The associations between different classes, and their multiplicities were also chosen correctly. Furthermore, DiagrammingAI identified the dependencies between classes, such as “Customer applies for a Loan Application”, “Organization reviews LoanApplication”. The generated diagram effectively depicts the intended behaviour of the classes by providing the operations. While the majority of operations DiagrammingAI included, are relevant for our system, a few operations important for the business logic were omitted (e.g choosing identification method for the Customer, or updating the status for the LoanApplication, etc).

All in all, I believe that this tool can be convenient for developing an initial sketch of a class diagram. Its strengths lie in identifying the basic operations, and relationships between classes based on a brief description. However, due to the lack of context, it can miss crucial domain-specific features, which can be considered a weakness. By iterating and constantly refining the prompt with details, the diagram provided by the DiagrammingAI can become increasingly similar to the actual diagram intended for the system. Although we did not update our model with the details provided by the tool, it was useful for validating that our model aligns with the standard conventions for UML class diagrams.