**System Description: Small Loan Application System**

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

**Task 1: Description**

The **Small Loan Application System** is designed to streamline the process of providing customers with small loans ranging from **500 to 15,000 EUR** for purchasing products at shops affiliated with an organization. The system ensures efficient loan application and approval processes by incorporating multiple user roles and a well-structured workflow.

**Classes, Attributes, and Relationships**

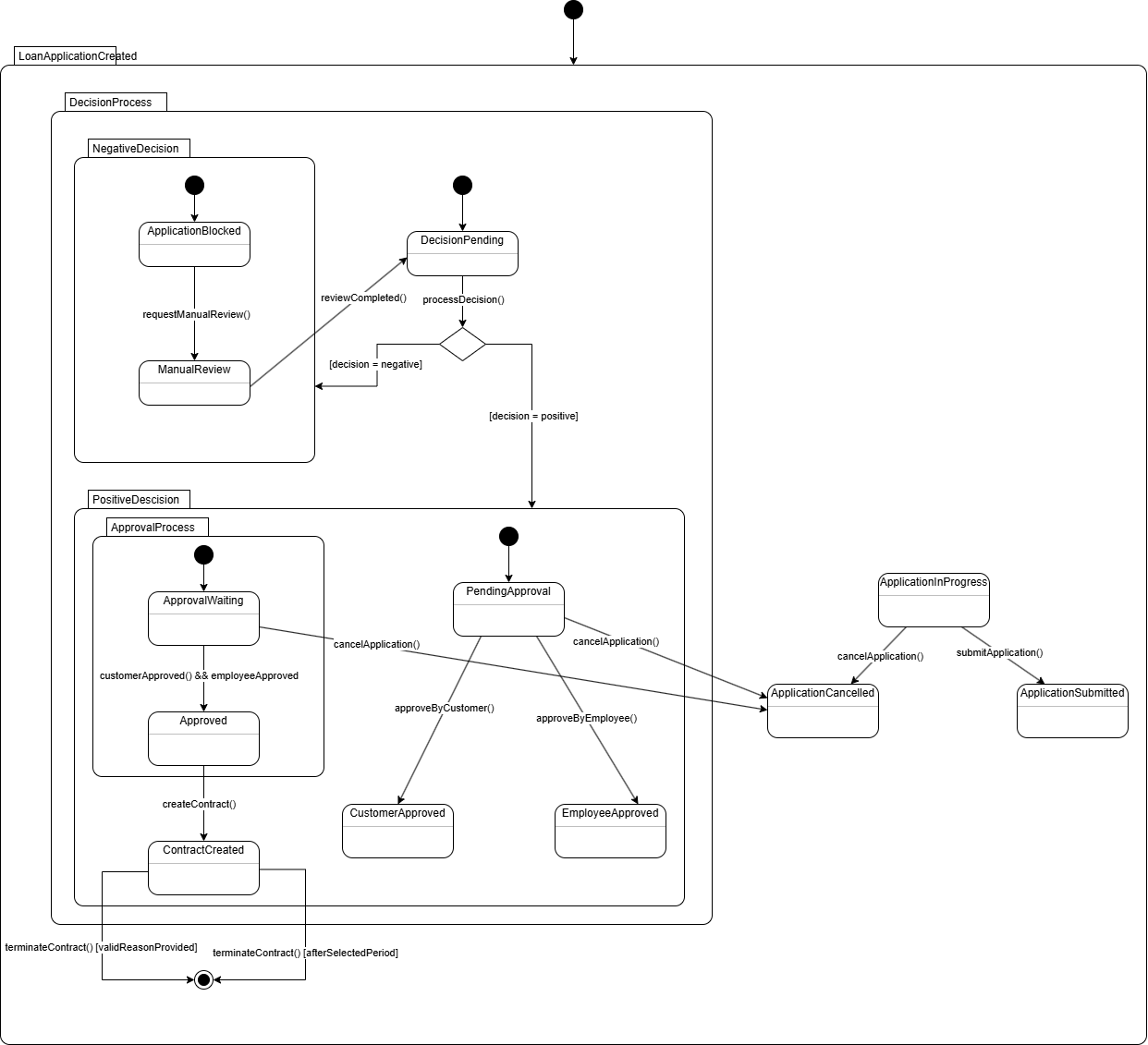
|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Attributes** | **Relationships** | **Actor Operations** |
| **Customer** | - CustomerID  - Identity Code  - Name  - Contact Information  - Gender  - Birthday  - Citizenship  - Occupation | - Can have multiple **Loan Applications** (1..n) | - Provides personal details and signs approved Loan Applications. |
| **Employee** | - EmployeeID  - Experience  - Position  - Shift | - Approves **Loan Applications** (1..n)  - Associated with one **Organization** (1..1) | - Manages manual reviews for applications with a negative DecisionStatus.  - Approves and signs Loan Applications. |
| **Loan Application** | - ApplicationID  - LoanAmount  - LoanPeriod  - Interest Rate  - Decision Status (positive/negative)  - Creation Date  - Product  - Customer  - CustomerSignature  - EmployeeSignature | - 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** | - ContractID  - Status  - TerminationDate  - Application  - Duration  - Amount  - Shop  - Start Date | - Originates from one **Loan Application** (1..1) |  |
| **Organization** | - OrganizationID  - Name  - Address  - Contact Info | - Owns multiple **Shops** (1..n)  - Employs multiple **Employees** (1..n) | - Facilitates automated loan decisions.  - Provides oversight and governance of Shops and Employees. |
| **Shop** | - ShopID  - Address  - ContactInfo | - Belongs to one **Organization** (1..1)  - Offers multiple **Products** (1..n)  - Facilitates multiple **Loan Applications** (1..n) | - Provides Product Information and Application Form Handling |
| **Product** | - ProductID  - Name  - Price  - Description  - Quantity | - Available at multiple **Shops** (1..n)  - Linked to multiple **Loan Applications** (1..n) |  |

**System Workflow:**

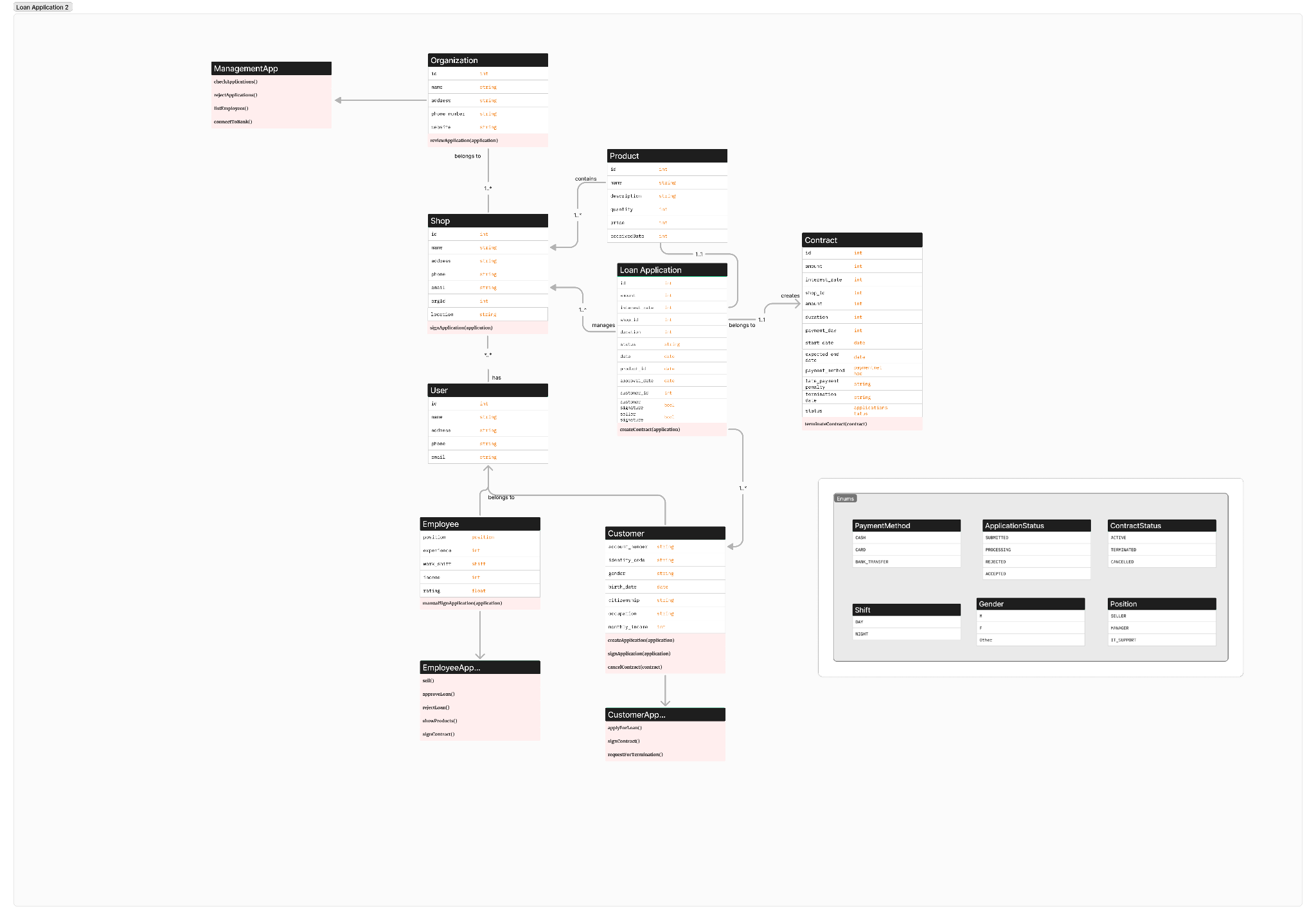
* Customer visits a Shop and selects a Product.
* If interested in financing, a Loan Application is created, incorporating data from the Customer, Product, Shop, and Organization, along with loan amount and period.
* The Loan Application undergoes an automatic decision process by the Organization, resulting 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. Both customer and employee can cancel the application during the waiting for approval process or pending approval process.
* Once approved by both parties, the Loan Application transitions into a Contract.
* Contract termination is allowed after the selected period or by providing a valid reason.

**Task 2: State chart diagram**

After slightly modifying the description of a system, a State chart diagram was created. Multiple composite states were used, such as: LoanApplicationCreated, DecisionProcess, NegativeDecision, PositiveDecision, ApprovalProcess.



**Task 3: Application class model**

****

<https://www.figma.com/board/pSn5QPRGnAveXV0f3bPlea/Loan-Application?node-id=0-1&node-type=canvas&t=WTsMtrwZQdWSXa8R-0>

**Task 4: Environment**

For designing my application class model, I chose Figma. The primary reason for this choice was that the initial class model assignment was done in Figma, making it convenient to extend and build upon. Additionally, Figma is user-friendly, with a lot of customization options that facilitate the creation of detailed and visually appealing diagrams. However, Figma does not support automatic code generation from the designed models.

**Observations on the Original Model**

The original model lacked user interactions, which I identified and addressed by adding the following components:

* ManagementApp: This app is designed to manage all shops, providing a centralized interface for shop management.
* EmployeeApp: This app allows employees to manage and sign loan applications for customers, streamlining the loan approval process.
* CustomerApp: This app enables customers to apply for loans and sign contracts to get loans for purchasing products, enhancing the user experience and accessibility.

I also made changes to the original models by adding and modifying some fields to enhance the functionality and accuracy of the system:

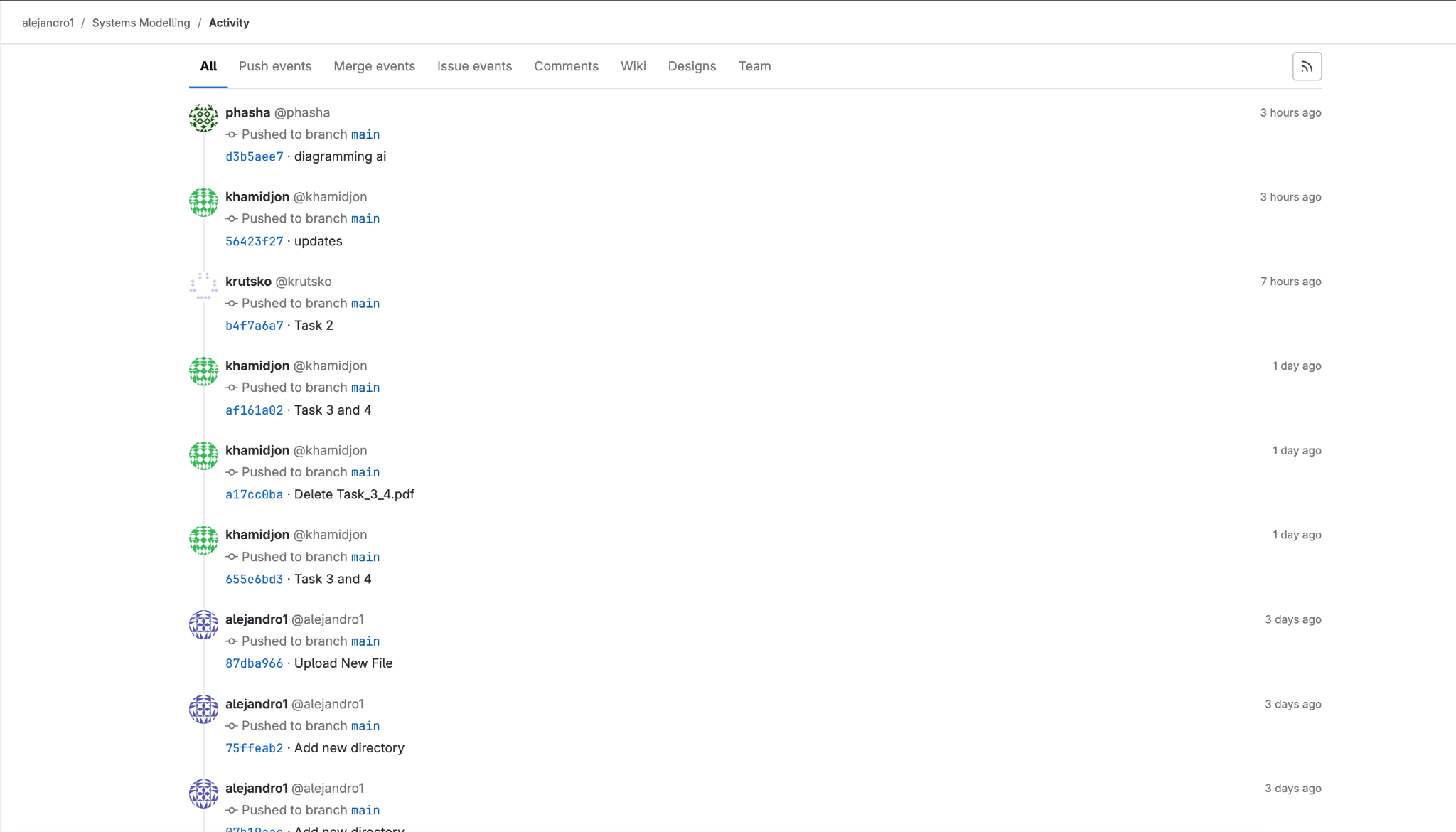
* Product model: Added received\_date to track when products are received, which is essential for inventory management and ensuring the timely processing of goods.
* Shop: Added location to record the physical location of each shop, which is crucial for managing multiple shop branches and optimizing logistics.
* LoanApplication: Changed organisation\_signature to seller\_signature to more accurately reflect the party responsible for approving the loan applications, ensuring clear accountability.
* Employee: Added income and rating to provide a comprehensive view of each employee's financial status and performance, aiding in better human resource management and decision-making.

For designing my statechart, I used draw.io. This tool is free, modern, and powerful, making it a good choice for creating detailed statecharts. Draw.io also offers integration with popular platforms like Atlassian, Notion, and Google Workspace, which helps in collaborative environments and improves productivity.

**Code Generation**

Figma does not support auto-generation of code from the designed models, which means that any translation from model to code must be done manually. This limitation can be a drawback as it requires additional effort to ensure that the code accurately reflects the designed model.

**Task 5: Evidence of Teamwork**

****

**Task 6: Explore and document Generative AI capabilities**

**Assignment AI**

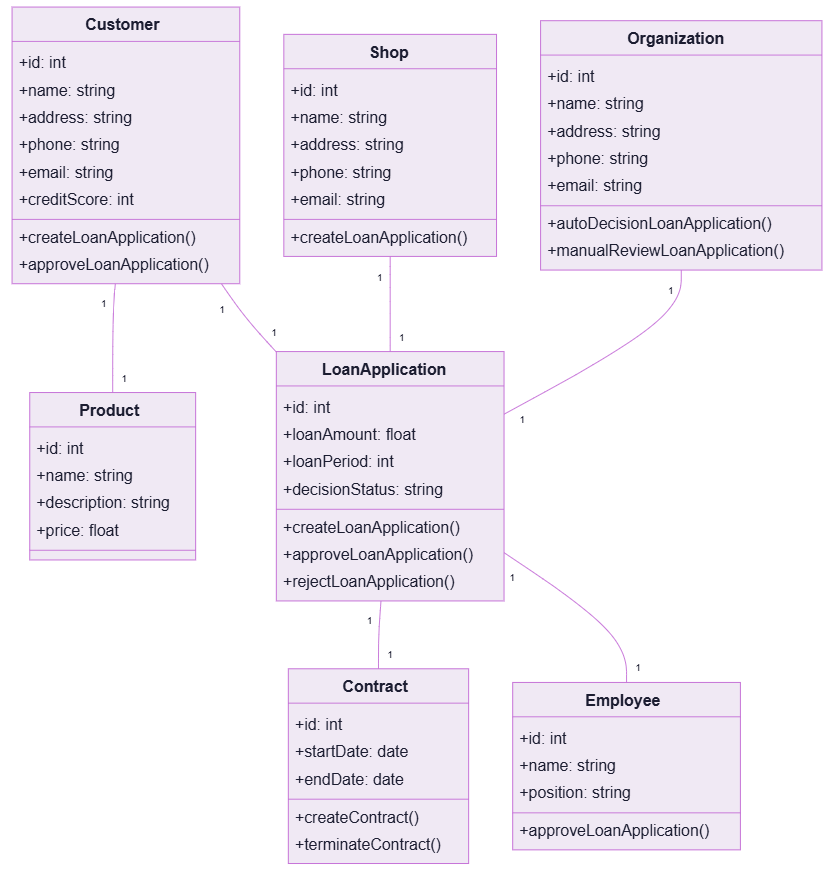
Assignment AI proved to be very easy to use. It has a wide variety of diagrams it can generate, and the UI is very dynamic. The resulting diagram is given with mermaid notation and can be saved and/or downloaded for free. Even though it has a subscription service, the free version does a good job, better than other AI options I have tried. It also includes an option to select the different complexity levels of the diagram between Beginner, Intermediate, and Advanced.

**Prompts Used**

The Small Loan Application System is designed to streamline the process of providing customers with small loans ranging from 500 to 15,000 EUR for purchasing products at shops affiliated with an organization. The system ensures efficient loan application and approval processes by incorporating multiple user roles and a well-structured 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.
* Contract termination is allowed after the selected period or by providing a valid reason

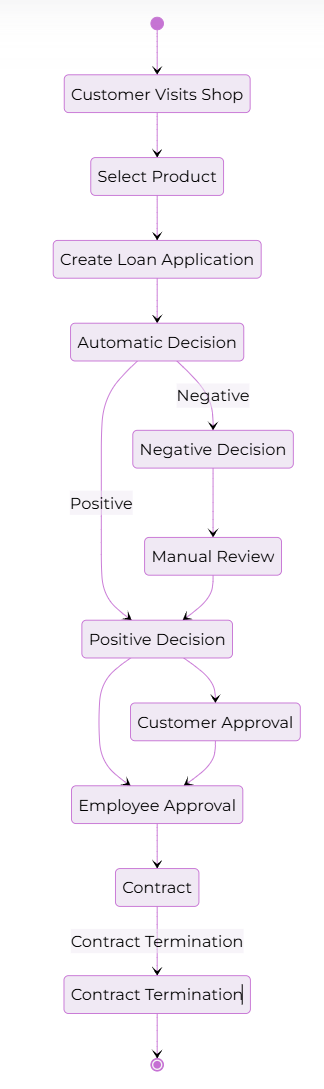
**Application Class Diagram**

****

**Comparison**

* The AI model did a good job when generating the classes. It got all of them except for the General User class.
* The attributes inside of the AI classes are fewer, but the essential ones are there.
* It is missing the signature and approval attributes, but it has a decision Status which might cover them in a general aspect.
* Multiplicity was also included, but a 1-1 multiplicity was used for all of the relationships between classes.
* Relationship direction and description was not included either in the AI diagram.
* Enums are also not used, even if asked specifically to use them.
* In general the attribute types were chosen correctly in general terms, as ids are ints, and the rest of the attributes are Strings or dates, which can be a real situation if no enums were used at all.

**State Diagram**

****

**Comparison**

* Even though Assignment AI states that it can generate state diagrams, it barely holds as one, it kind of also could be considered a very rough activity diagram.
* In AI diagram, no frames are used, and many of the states are not corresponding to the Loan Application
* No events used
* No activities or triggers
* Way less complex, without history
* In general a very poor state diagram. It could be used to have a very rough idea of what is expected, but it's well made at any point

**Miro AI**

I chose Miro and Miro AI for creating class and state diagrams because it is a widely used, trusted tool in businesses for collaborative and visually engaging workflows. Its intuitive interface and versatility make it an ideal platform for structuring complex systems with ease. The integration of Miro AI further enhances productivity by automating parts of the design process, providing intelligent suggestions, and ensuring a smoother workflow, especially for brainstorming and initial drafts.

**Prompt Used**

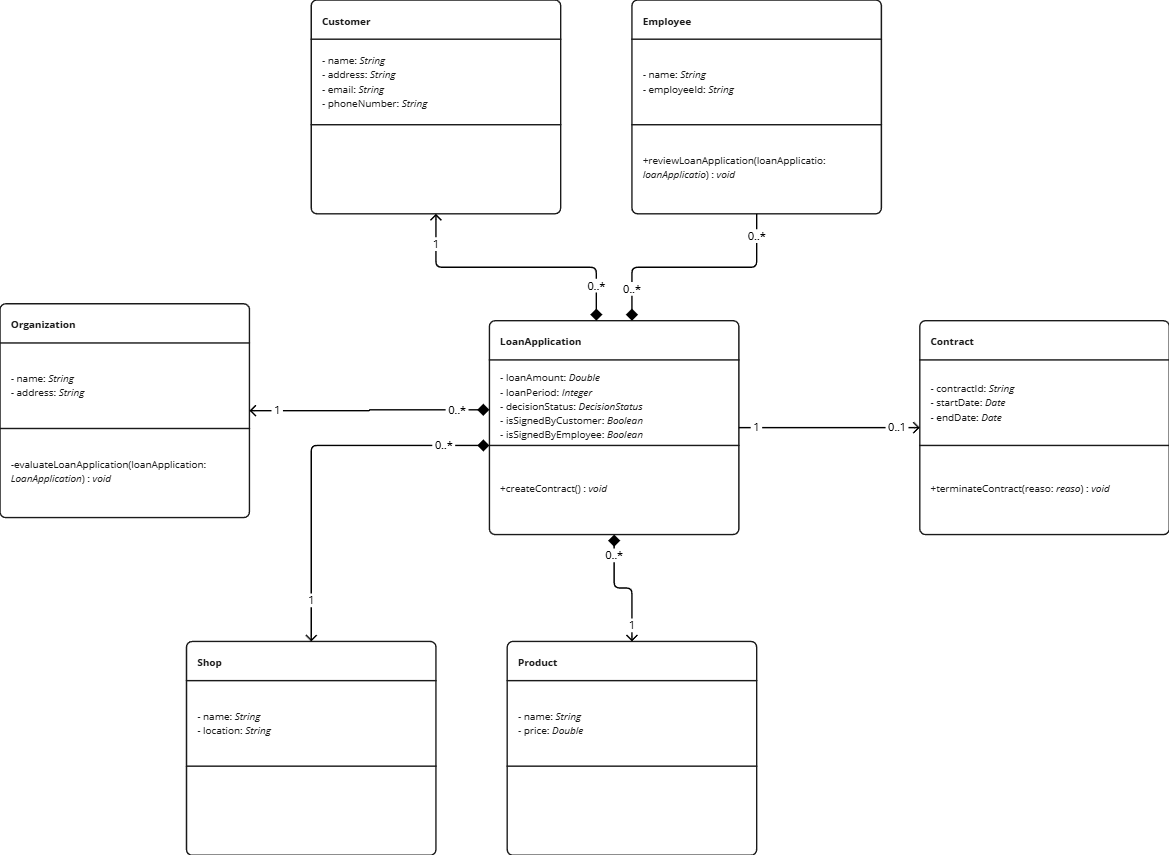
Make me a class diagram with aggregation, multiplicity, and association direction for

A Small Loan Application System is designed to streamline the process of providing customers with small loans ranging from 500 to 15,000 EUR for purchasing products at shops affiliated with an organization. The system ensures efficient loan application and approval processes by incorporating multiple user roles and a well-structured workflow.

**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.
* Contract termination is allowed after the selected period or by providing a valid reason

**Application Class Diagram**

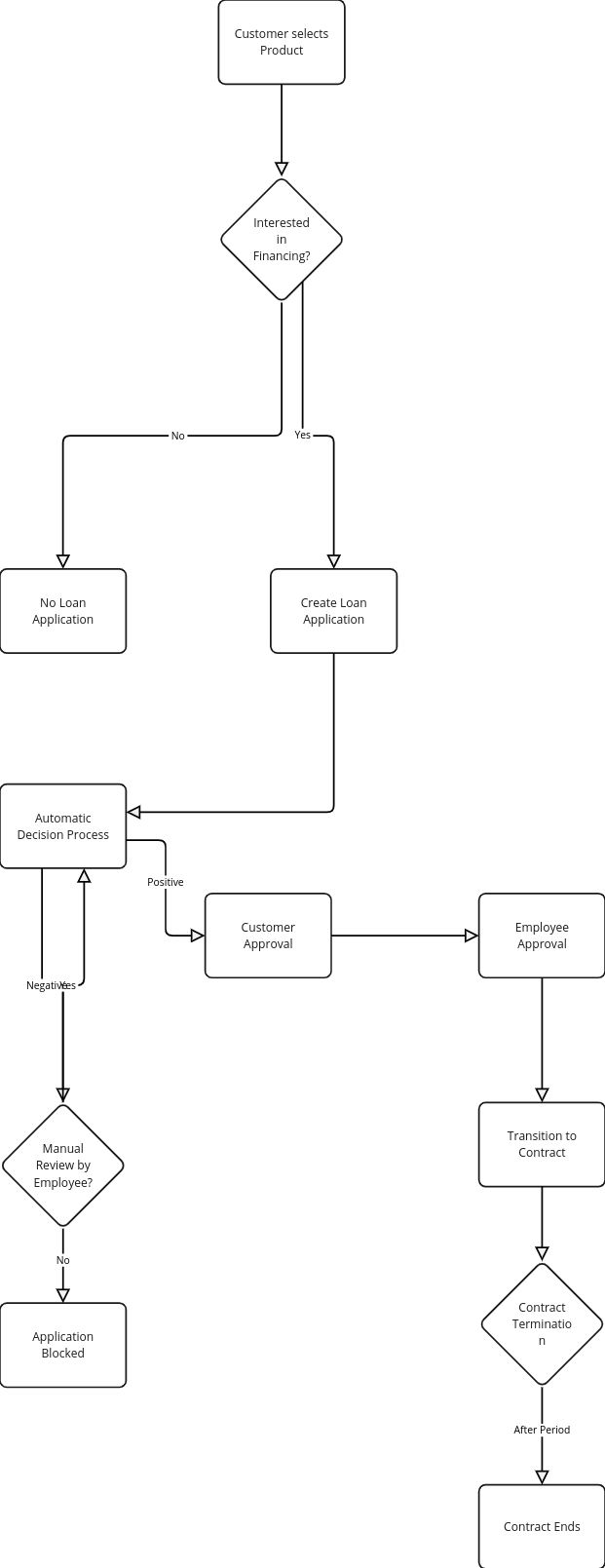


**Comparison**

* AI once again got almost all requested Classes.
* Multiplicity is also asserted and directional arrows were included, but no name.
* Almost all relations are composition, which in this case is incorrect since Customer, Shop, Employee and other classes are not owned by LoanApplication, and would not be deleted if the LoanApplication were to be deleted.
* Operations in the AI model from Miro are fewer, but they are in the correct classes.
* Customer is missing all of its operations
* No relationship directions
* Some operations missing

**State Diagram**

Miro AI doesn’t have a state diagram generator, but it does have a flowchart. It's not the same but it’s worth analyzing.



**Comparison**

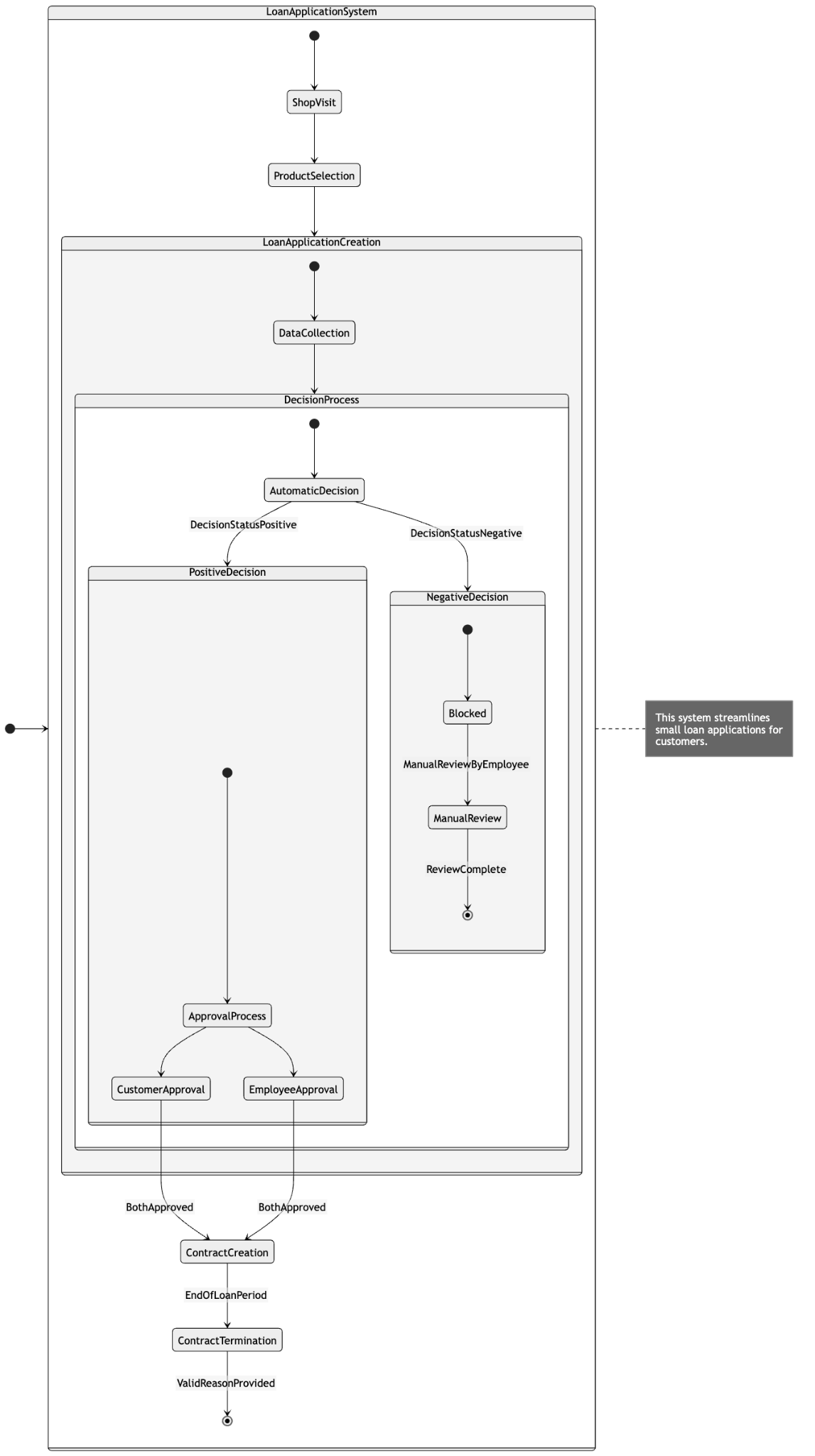
Since it is not a state diagram directly, it is understandable that there are many details missing like the event calls, start and end symbols and more. There are some flows that were not in the Assignment AI diagram, such as application blocked, that could be used. By combining both of these diagrams a more complete state diagram could be developed

**Diagramming AI**

DiagrammingAI is a GPT-based GenAl 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. It allows creating various state diagrams as well - basic, HSFM, concurrent, etc.

**State Diagram**

Given the same prompt as in the above cases, DiagrammingAI managed to produce an acceptable general picture of the system in terms of events and states included in it. As we can see, the loan approval process was understood correctly, although represented a bit differently (e.g no separate state for pending applications). The chart has composite states (structured differently than in our diagram) and guards. Additionally, guards for contract termination were aligned with those identified by us. Moreover the diagram includes triggers, some of which are the same as in our chart. All in all, diagramming AI correctly identified the main states and events in the contract creation process, and produced a reasonable statechart.

****

**Conclusion**

AI diagram model generators today can produce impressive rough drafts of diagrams, making them valuable tools for quickly visualizing ideas. However, they are not perfect in any area and still require significant human intervention to refine and adapt the output to specific needs. The diagrams they generate tend to be simplistic, often missing key entities, relationships, or nuances of the system. Even when provided with more detailed descriptions, these tools struggle to grasp the big picture, frequently defaulting to a straightforward approach that oversimplifies complex systems.

This task helped to refactor the description from Task #1 little bit, and this exercise demonstrated that diagrams similar to the ones we have were successfully generated from our description. While these tools are great for creating starting points, they still rely heavily on human oversight to reach the level of depth and precision required for complex systems.