Ecommerce
BOOK’S

JONATHAN JAVIER CASTELLANOS BRAVO

OBJECTIVES

The primary objectives of this project include:

- Provide a user-friendly web interface for book shopping.
- Implement a secure authentication mechanism for users.
- Allow inventory and product management.
- Create a sales register.
- Improve sales volume with solution

SCOPE

FEATURES

- User registration and authentication
- Product management (add books)
- Inventory control
- Sales register
- Basic interfaces

EXCLUDES ADVANCED FEATURES

- Payment gateway integration
- User reviews or recommendations
- Order tracking
- Full e-commerce analytics

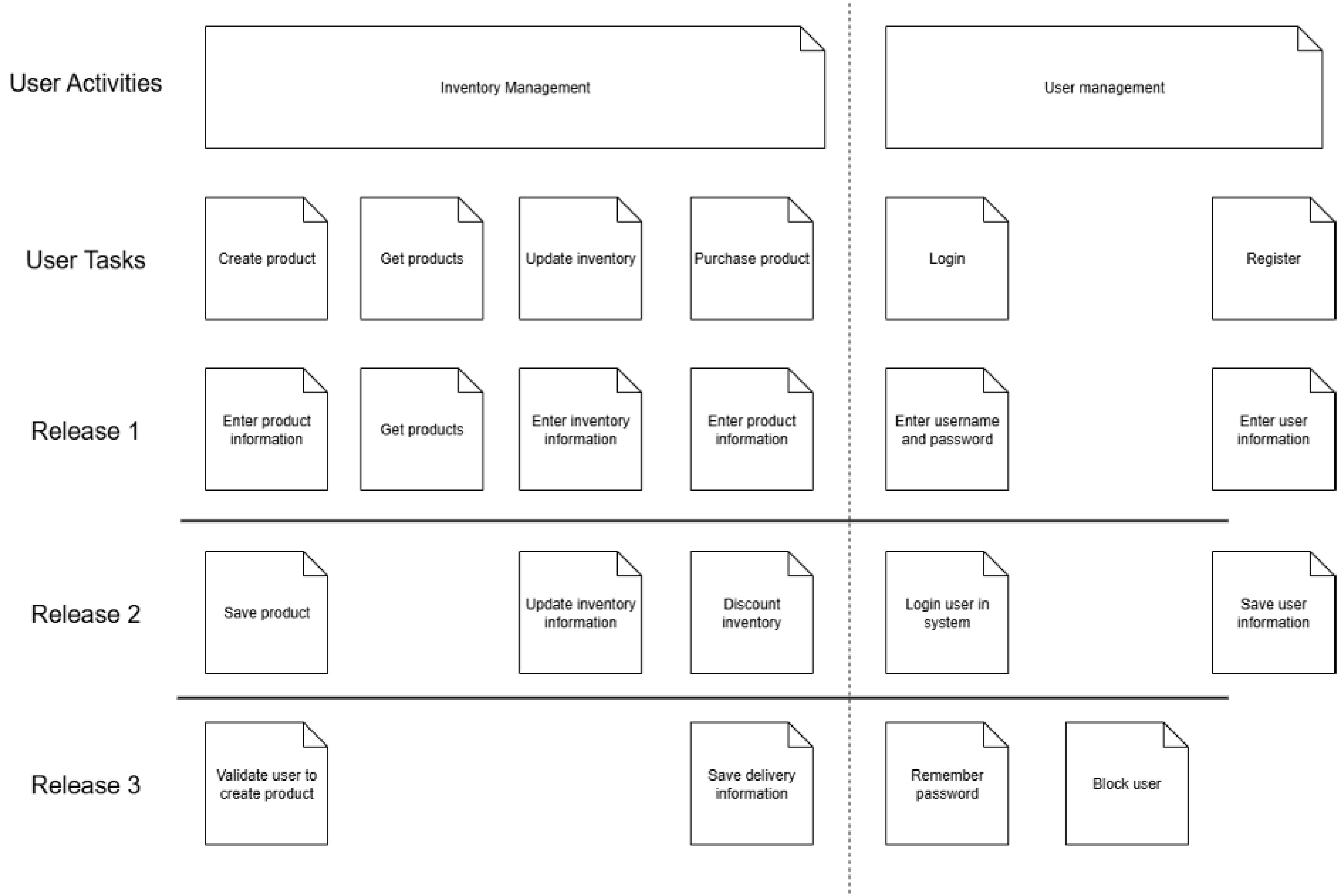
ASSUMTIONS

- The application is intended to serve a small business with max 100 users per day.
- The system will be deployed in a controlled environment with limited traffic spikes.
- Only one administrator manages the backend operations.
- Data used for testing represents realistic scenarios.
- Security mechanisms are limited to JWT-based authentication for the MVP stage.
- The system is expected to run on a single server instance

LIMITATIONS

TEAM SIZE:	TIME	UI/UX:	EXTERNAL INTEGRATIONS:
The project was developed entirely by a single developer, which constrained the imple mentation timeline and scope.	The available time allowed only the development of essential modules for the MVP. Features such as scalability optimization are planned for later phases	Frontend design was kept minimal and functional, with limited focus on user experience aesthetics.	No third-party payment or shipping services were included in this version.

VISUAL STORY MAPPING

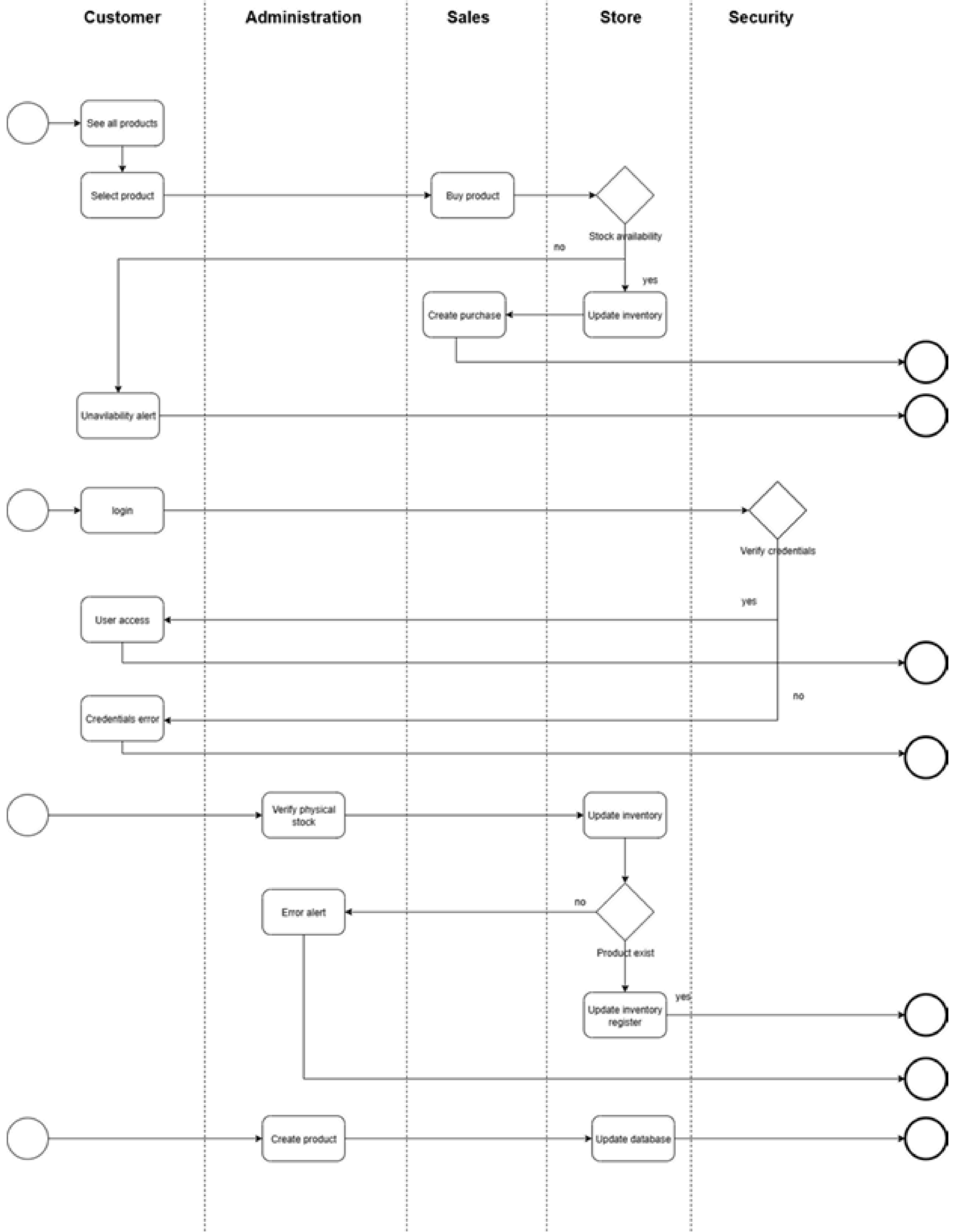


User Activities	Inventory Management				User management		
User Tasks	Create product	Get products	Update inventory	Purchase product	Login		Register
Release 1	Enter product information	Get products	Enter inventory information	Enter product information	Enter username and password		Enter user information
Release 2	Save product		Update inventory information	Discount inventory	Login user in system		Save user information
Release 3	Validate user to create product			Save delivery information	Remember password	Block user	

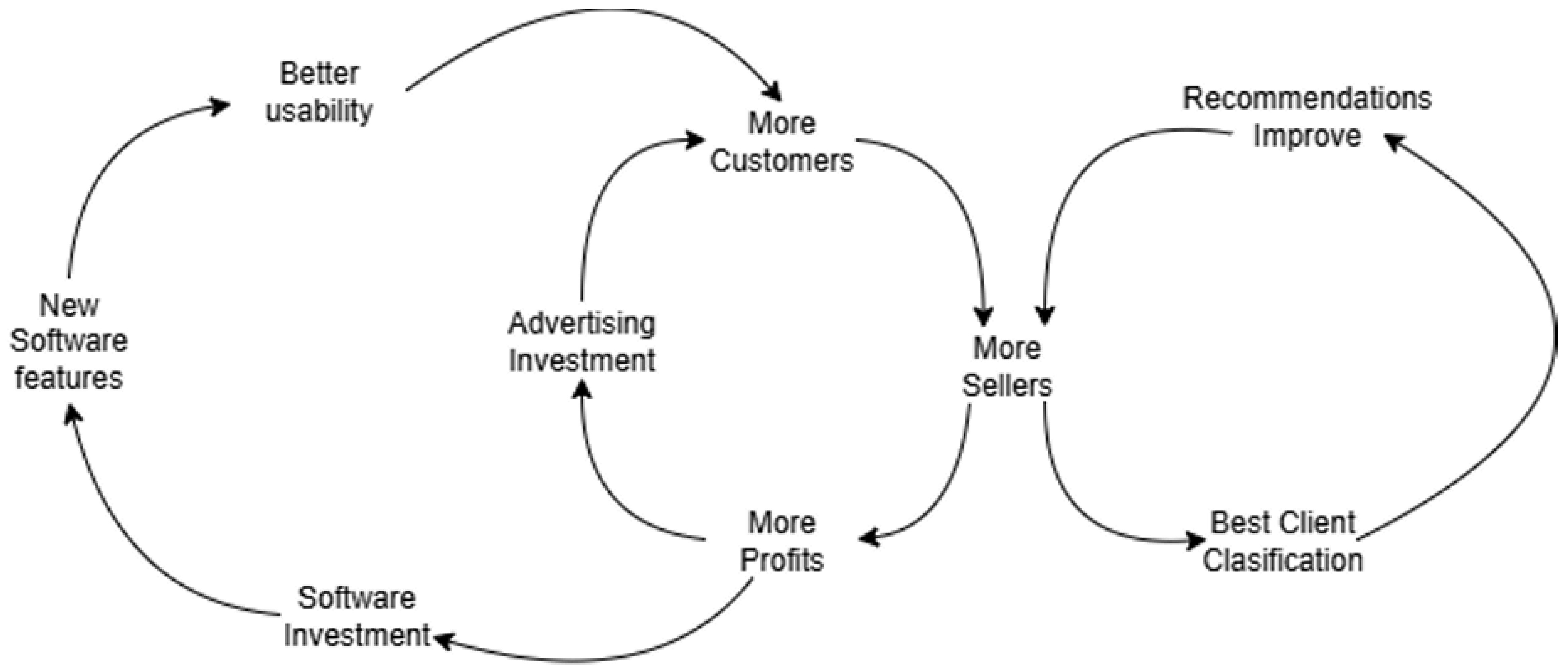
USER STORIES

- As an administrator, I want tocreate a new product by entering its information, so that it can be available in the syste.
- As an administrator, I want to update the inventory of a product so that the available quantity reflects the actual stock.
- As a logged inuser, I want to purchase products from the system so that they can be shipped to my address.
- As a new user, I want to register in the system so that I can make purchases.
- As a registered user, I want to log into the system so that I can access it.

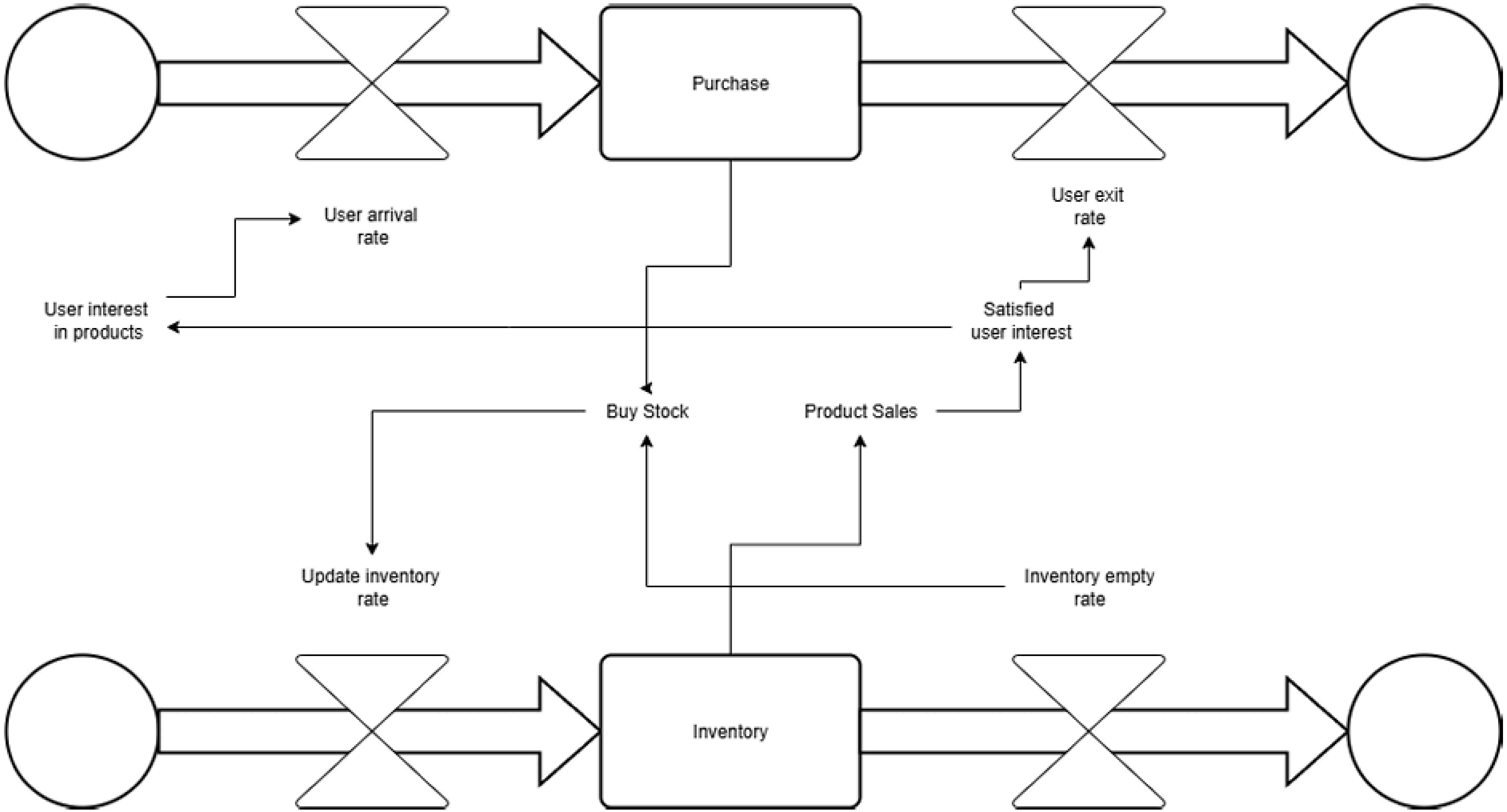
BUSINESS PROCESS MODELING (BPMN)



CAUSAL LOOP DIAGRAM



STOCK AND FLOW DIAGRAM



ARCHITECTURE DESIGN

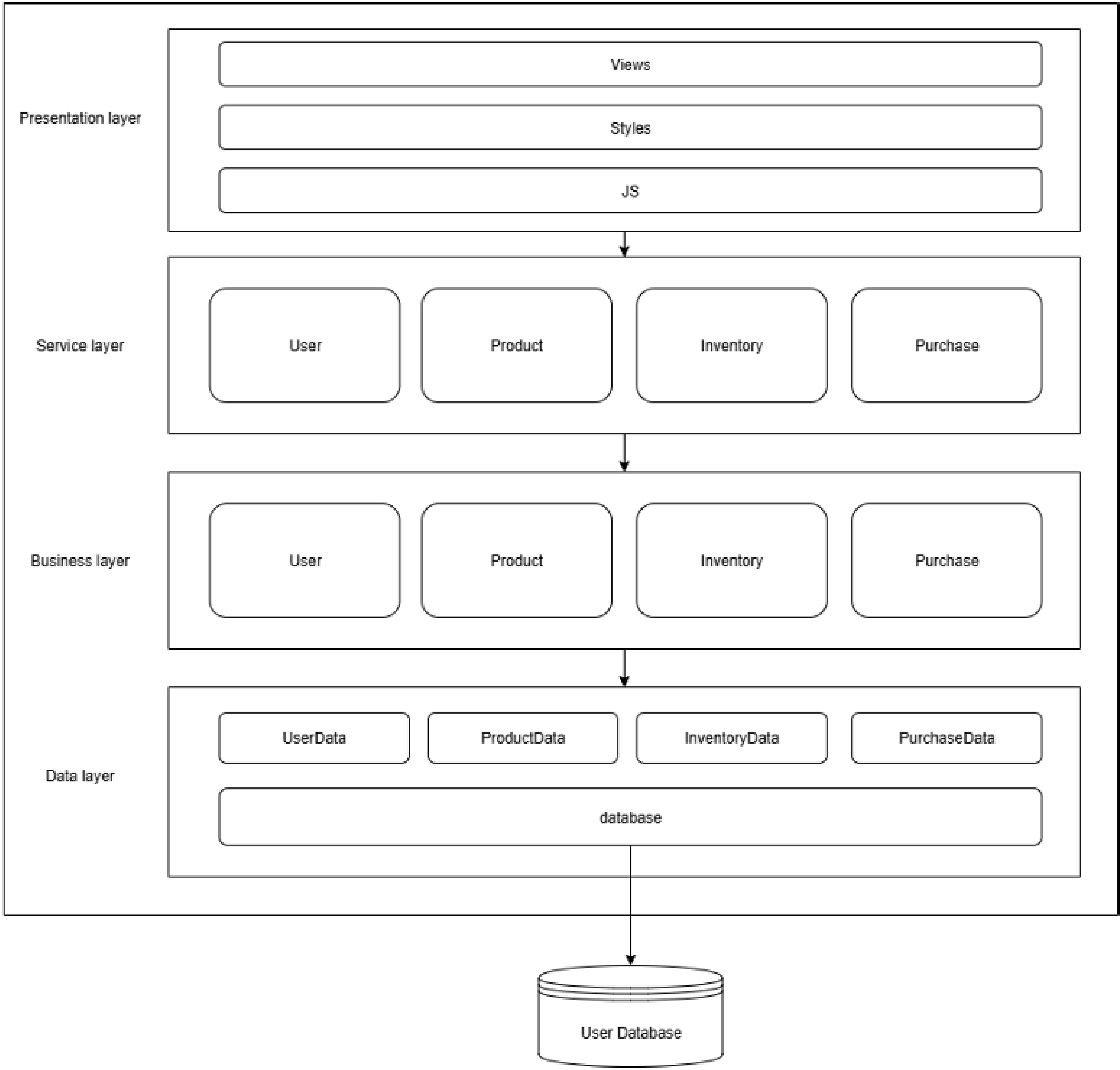
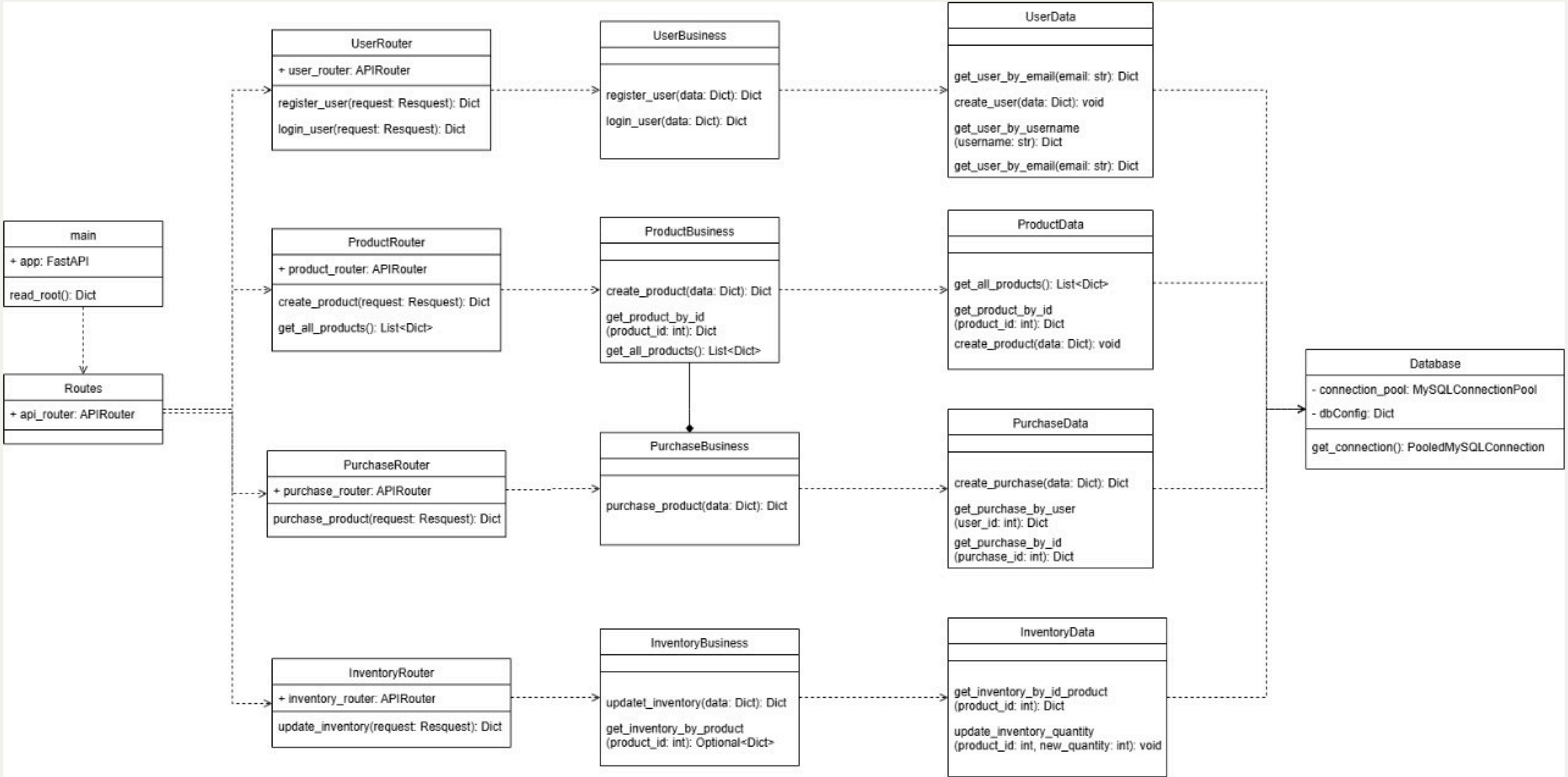


Figure 5: Architecture Diagram

CLASS DIAGRAM



main
+ app: FastAPI
read_root(): Dict

Routes
+ api_router: APIRouter

UserRouter
+ user_router: APIRouter
register_user(request: Resquest): Dict
login_user(request: Resquest): Dict

ProductRouter
+ product_router: APIRouter
create_product(request: Resquest): Dict
get_all_products(): List<Dict>

PurchaseRouter
+ purchase_router: APIRouter
purchase_product(request: Resquest): Dict

InventoryRouter
+ inventory_router: APIRouter
update_inventory(request: Resquest): Dict

UserBusiness
register_user(data: Dict): Dict
login_user(data: Dict): Dict

ProductBusiness
create_product(data: Dict): Dict
get_product_by_id
(product_id: int): Dict
get_all_products(): List<Dict>

PurchaseBusiness
purchase_product(data: Dict): Dict

InventoryBusiness
updatet_inventory(data: Dict): Dict
get_inventory_by_product
(product_id: int): Optional<Dict>

UserData
get_user_by_email(email: str): Dict
create_user(data: Dict): void
get_user_by_username
(username: str): Dict
get_user_by_email(email: str): Dict

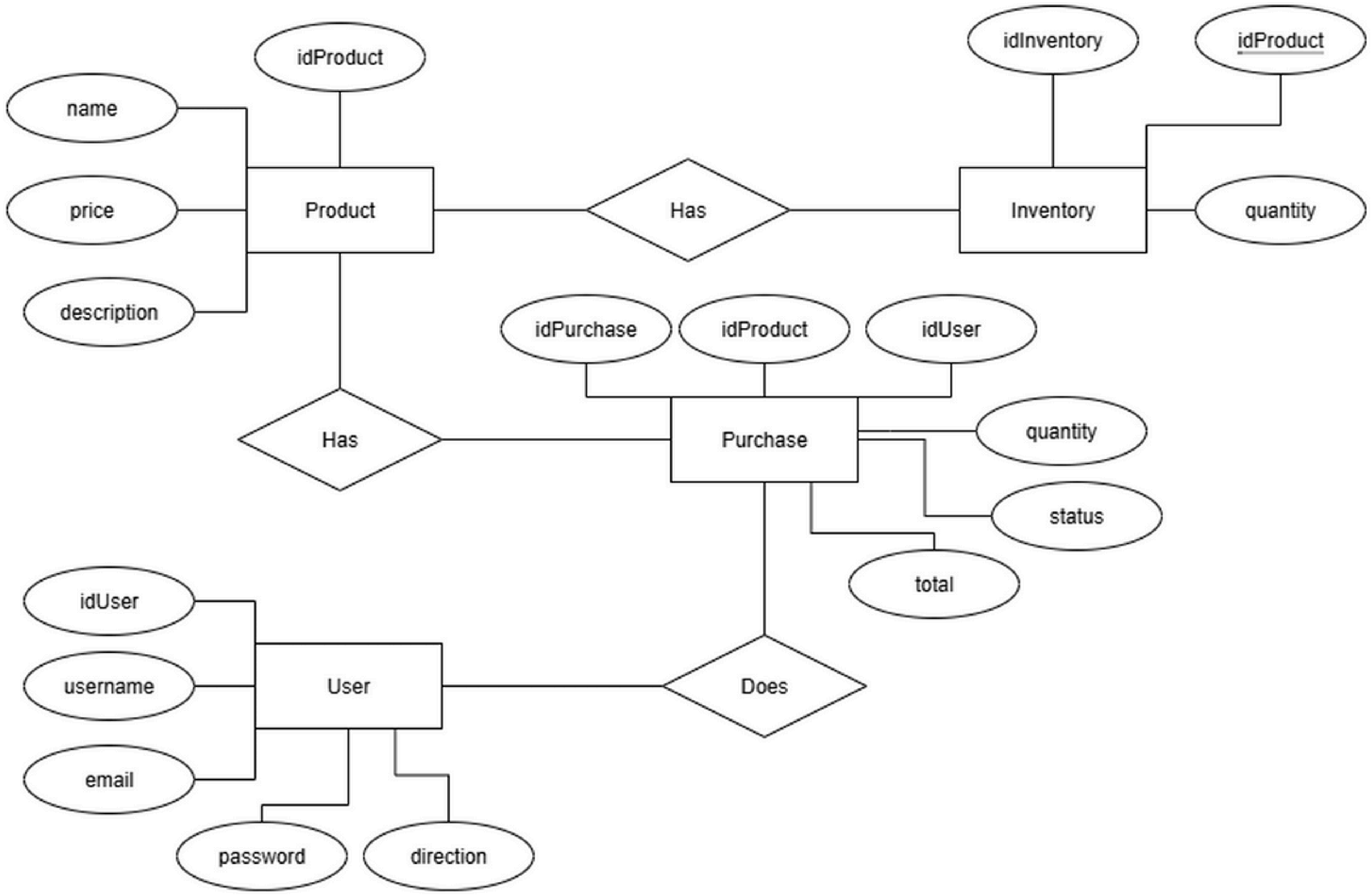
ProductData
get_all_products(): List<Dict>
get_product_by_id
(product_id: int): Dict
create_product(data: Dict): void

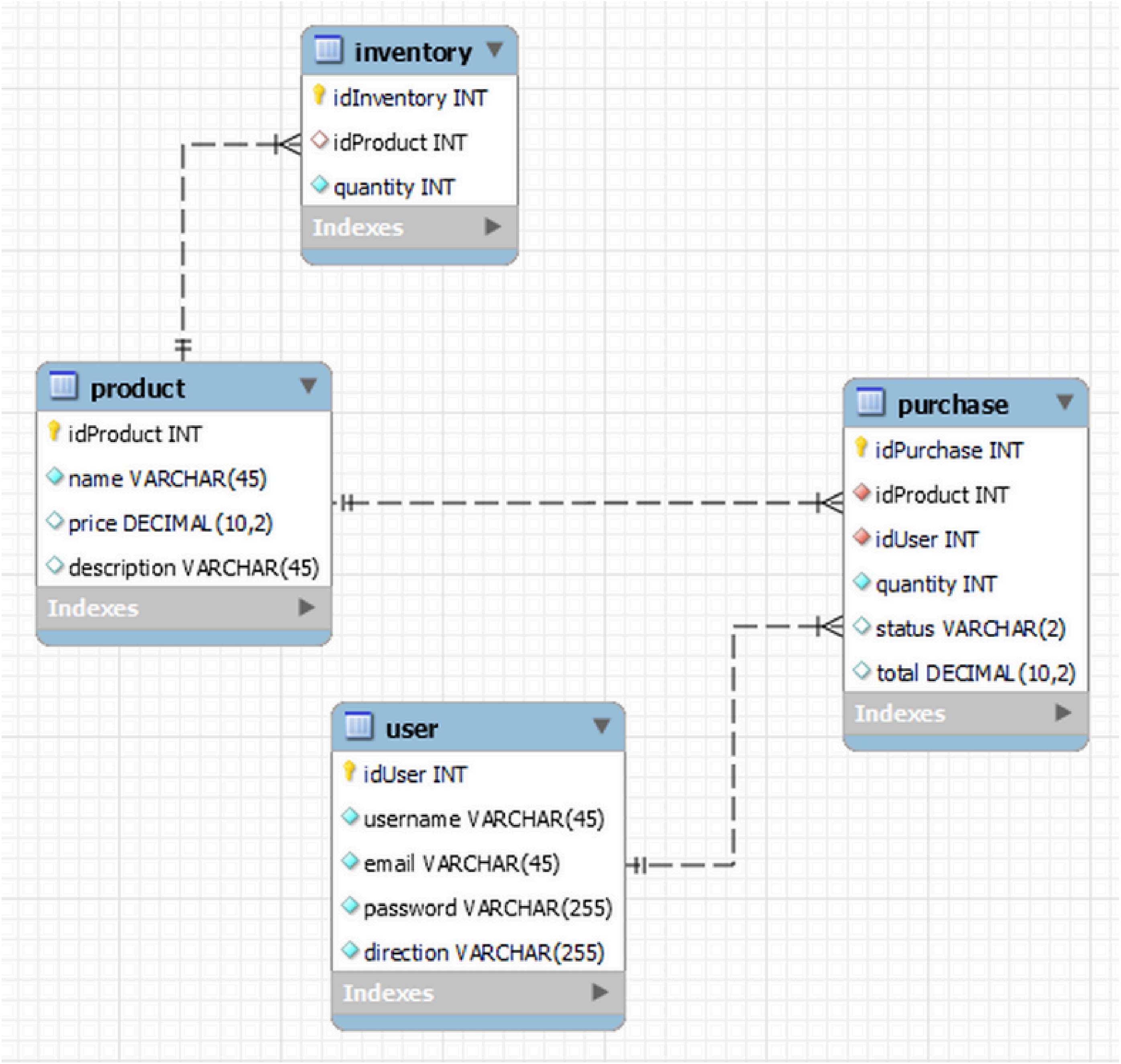
PurchaseData
create_purchase(data: Dict): Dict
get_purchase_by_user
(user_id: int): Dict
get_purchase_by_id
(purchase_id: int): Dict

InventoryData
get_inventory_by_id_product
(product_id: int): Dict
update_inventory_quantity
(product_id: int, new_quantity: int): void

Database
- connection_pool: MySQLConnectionPool
- dbConfig: Dict
get_connection(): PooledMySQLConnection

DATA MODELING





RESULTS

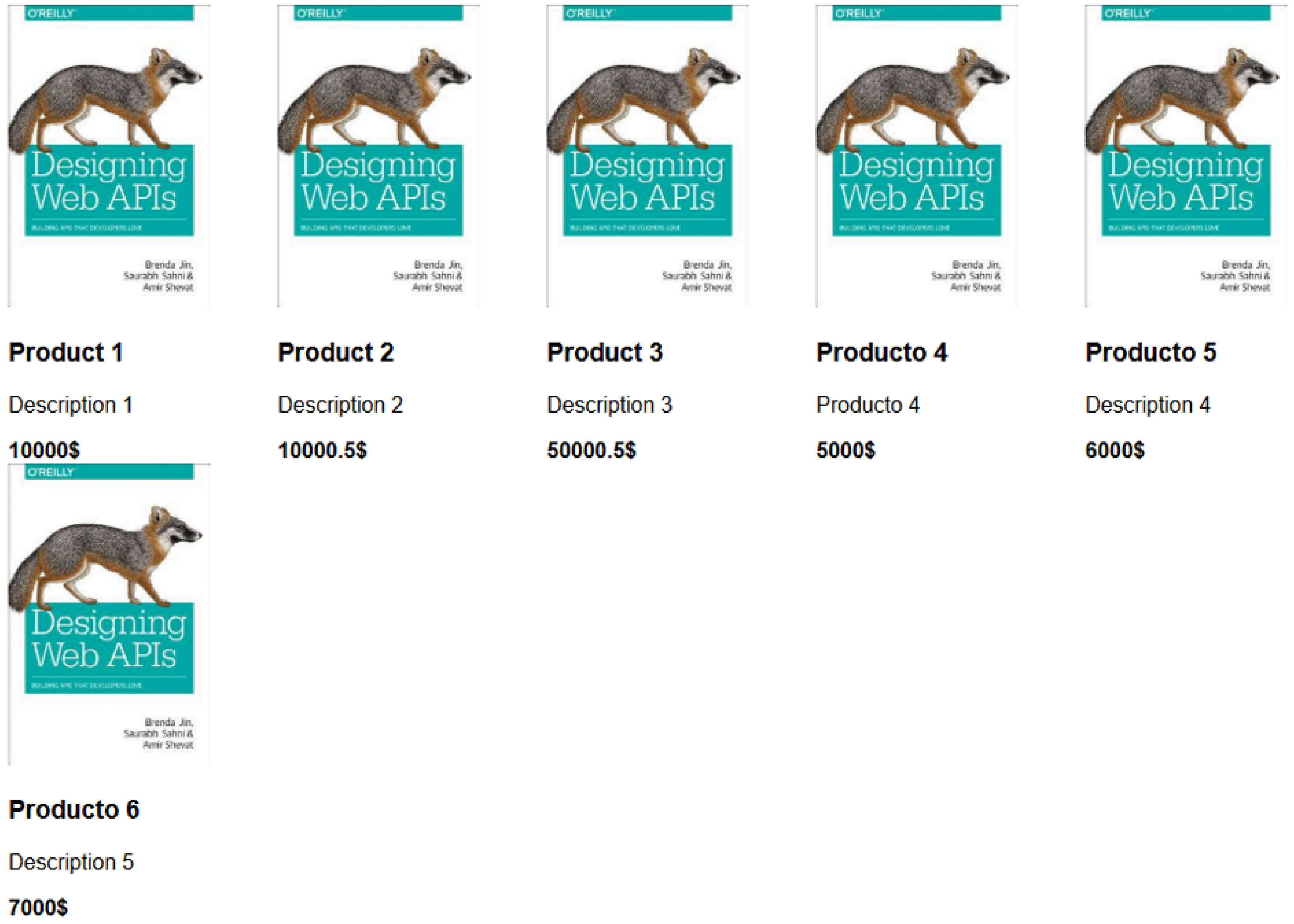
- Afully functional prototype of the ecommerce application was developed.

- Integration tests confirmed the correctness of:

– User creation and login flows

– Book creation and inventory management

– Sales recording endpoint



Product 1

Description 1

10000$

Product 2

Description 2

10000.5$

Product 3

Description 3

50000.5$

Producto 4

Producto 4

5000$

Producto 5

Description 4

6000$

Producto 6

Description 5

7000$

Unit Test

TESTS WERE CARRIED OUT FOR EACH FUNCTION WITHIN THE CODE. FOUR CONDITIONS WERE CONSIDERED FOR EVERY TEST:

1. Successful case – when the function executes as expected.
2. Incorrect parameters – when invalid or unexpected inputs are provided.
3. Missing parameters – omission of one or more required parameters.
4. Database transaction error – when the process fails due to issues interacting with the database.

THE TEST DETAILS ARE IN THE REPORT.

Integration test

TESTS WERE PERFORMED FOR EACH FUNCTIONALITY EXPOSED BY THE API, CONSIDERING THE FOLLOWING THREE SCENARIOS IN ALL CASES:

1. Successful case – correct execution of the functionality.
2. Incorrect parameters – submission of invalid data or data in an unexpected format.
3. Missing parameters – omission of one or more required parameters.

THE DETAILED TEST RESULTS ARE AVAILABLE IN THE REPORT.

System **Test**

TESTS WERE PERFORMED FOR EACH FUNCTIONALITY EXPOSED BY THE API, CONSIDERING THE FOLLOWING THREE SCENARIOS IN ALL CASES:

1. Login
2. Purchases
3. Product queries

THE DETAILED TEST RESULTS ARE AVAILABLE IN THE REPORT.

Discussion

The integration test results show a structured and thorough evaluation of API functionality and reliability. The Postman collection is well organized by modules, covering successful cases, error handling, and edge cases, confirming correct software validation. The first phase met expectations, though some planned features remain to be implemented to fully meet business needs.