VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
SOFTWARE ENGINEERING

Program System Engineering

Laboratory work 2

Team OmegaPoint

Made by:	3rd year students
	Benas Budrys
	Šarūnas Griškus
	Rugilė Vasaitytė
	Deividas Baltuška
Supervisor:	Vasilij Savin

Vilnius – 2023

CONTENTS

1. INTRODUCTION 2

2. HIGH-LEVEL SYSTEM COMPONENTS 3

3. ENDPOINTS EXPOSED FOR USER CONSUMPTION 5

4. OPERATIONAL ENVIRONMENT 9
   4.1. Geography 9
   4.2. Businesses 9
   4.3. Usage patterns 9
   4.4. User load 9
   4.5. Write-Read ratio 10
   4.6. Internalization 10
   4.7. Law 10
   4.8. Sensitive data 10
   4.9. Integration and dependencies with other systems 10

5. NON-FUNCTIONAL REQUIREMENTS 11
   5.1. Performance 11
   5.2. Legal requirements 12
   5.3. Usage context 13
   5.4. User experience 13
   5.5. Infrastructure, deployment 13

6. SCALING 14
   6.1. Regional sharding 14
   6.2. Sharding by client 14
   6.3. DNS load balancing 14
   6.4. Horizontally scalable server instances 14
   6.5. Internal load balancing 14

7. SYSTEM DESIGN CHANGES FROM PREVIOUS DOCUMENT 15

1. Introduction

The primary objective for this document is to prepare a technical documentation for the POS system implementation. Contained within the sections is a breakdown of the high-level system components, the endpoints exposed, the non-functional requirements, considerations related to scaling, and the API contract in the form of OpenAPI specification.

2. High-level system components

Figure 1 shows the high-level system component overview. Peripheral devices and e-commerce are the only publicly visible components, access to other components is granted only to certain employees based on their roles and associated permissions. The terminal acts as an intermediary device, that collects data from all peripheral devices, retrieves data from the PoS System and communicates information between the two when necessary. E-Commerce component represents the interface for customers online, it directly accesses the data from the server. The server(-s) hold PoS software, that includes Payments, User Management, Company Management, Inventory Management, Service Management, Order Management. Additional integrations for the PoS system may be included as necessary.

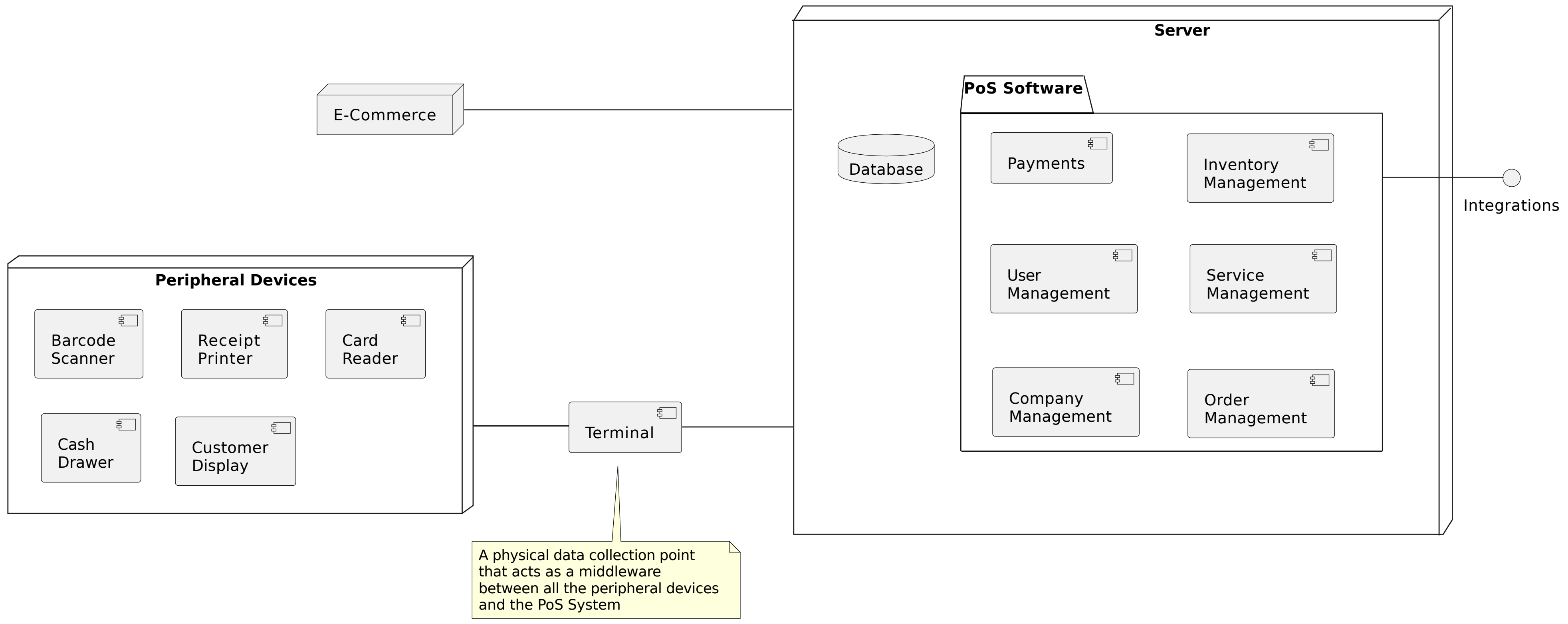

Figure 1 High-level system components

3. Endpoints exposed for user consumption

This section will contain brief descriptions of our main endpoint groups. For more detailed information, please refer to the API contract provided with this document.

Users API
CRUD - managed by managers, other users can manage their own data
GET /cinematic/{company_id}/users/
GET /cinematic/{company_id}/users/{user_id}
POST /cinematic/{company_id}/users
PUT /cinematic/{company_id}/users/{user_id}
DELETE /cinematic/{company_id}/users/{user_id}
Sign-in and reset the password - accessible for all unauthorized users
POST /cinematic/{company_id}/users/sign-in
Resetting of password logic should be covered on client-side applications, as it requires proving that you are surely the profile owner.
Loyalty points - managed by managers and employees, users can access their own data
GET /cinematic/{company_id}/users/{user_id}/loyalty
PUT /cinematic/{company_id}/users/{user_id}/loyalty
Loyalty offers CRUD - managed by managers and employees, users can access their own data
GET /cinematic/{company_id}/loyalty/
GET /cinematic/{company_id}/loyalty/{offer_id}
POST /cinematic/{company_id}/loyalty/
PUT /cinematic/{company_id}/loyalty/{offer_id}
DELETE /cinematic/{company_id}/loyalty/{offer_id}
Loyalty offer redeeming - accessible by users only
POST /cinematic/{company_id}/loyalty/{offer_id}/redeem
Roles management CRUD - managed by managers only
GET /cinematic/{company_id}/roles
GET /cinematic/{company_id}/roles/{role_id}
POST /cinematic/{company_id}/roles
PUT /cinematic/{company_id}/roles/{role_id}
DELETE /cinematic/{company_id}/roles/{role_id}

Orders API
CRUD - managed by managers and employees, customers can manage only their own data
GET /cinematic/{company_id}/orders
GET /cinematic/{company_id}/orders/{order_id}
POST /cinematic/{company_id}/orders
PUT /cinematic/{company_id}/orders/{order_id}
DELETE /cinematic/{company_id}/orders/{order_id}

Edit itemOrders in your order – accessible by managers, employees and users

POST /cinematic/{company_id}/orders/{order_id}/itemOrders

DELETE /cinematic/{company_id}/orders/{order_id}/itemOrders/{item_order_id}

Order status – managed by managers and employees only

PUT /cinematic/{company_id}/orders/{order_id}/status

Order discounts – accessible by managers only

POST /cinematic/{company_id}/orders/{order_id}/discounts

Self-assign an order – accessible by managers and employees only

PUT /cinematic/{company_id}/orders/{order_id}/assign

Get assignable item orders – accessible by managers and employees only

GET /cinematic/{company_id}/itemOrders

GET /cinematic/{company_id}/itemOrders/{item_order_id}

Item order status – managed by managers and employees only

PUT /cinematic/{company_id}/itemOrders/{item_order_id}/status

Self-assign an item order – managed by managers and employees only

PUT /cinematic/{company_id}/itemOrders/{item_order_id}/assign

Services API

CRUD – managed by managers and employees, customers can access only

GET /cinematic/{company_id}/services

GET /cinematic/{company_id}/services/{service_id}

POST /cinematic/{company_id}/services

PUT /cinematic/{company_id}/services/{service_id}

DELETE /cinematic/{company_id}/services/{service_id}

Service discounts – accessible by managers only

POST /cinematic/{company_id}/services/{service_id}/discounts

Appointments CRUD – managed by managers and employees, customers can access only their own data

GET /cinematic/{company_id}/appointments

GET /cinematic/{company_id}/appointments/{appointment_id}

POST /cinematic/{company_id}/appointments

PUT /cinematic/{company_id}/appointments/{appointment_id}

DELETE /cinematic/{company_id}/appointments/{appointment_id}

Items API

CRUD – managed by managers and employees, customers can manage only their own data

GET /cinematic/{company_id}/items

GET /cinematic/{company_id}/items/{item_id}

POST /cinematic/{company_id}/items

PUT /cinematic/{company_id}/items/{item_id}

DELETE /cinematic/{company_id}/items/{item_id}

Item discounts - accessible by managers only

POST /cinematic/{company_id}/items/{item_id}/discounts

Item options CRUD - managed by managers only, employes and users can access data

GET /cinematic/{company_id}/itemOptions/

GET /cinematic/{company_id}/itemOptions/{item_option_id}

POST /cinematic/{company_id}/itemOptions

PUT /cinematic/{company_id}/itemOptions/{item_option_id}

DELETE /cinematic/{company_id}/itemOptions/{item_option_id}

Inventory management CRUD - managed by managers and employees only

GET /cinematic/{company_id}/inventory

GET /cinematic/{company_id}/inventory/{inventory_id}

POST /cinematic/{company_id}/inventory

PUT /cinematic/{company_id}/inventory/{inventory_id}

DELETE /cinematic/{company_id}/inventory/{inventory_id}

Low inventory stock notifications - should be implemented on client-side application.

Payments API

Accept a payment - managed by managers and employees only, users can access their own data

POST /cinematic/{company_id}/payments

Generate receipts - managed by managers and employees, users can access their own data

GET /cinematic/{company_id}/payments/{payment_id}/receipts

Handle refunds and void payments - accessible by managers and employees only

POST /cinematic/{company_id}/payments/{payment_id}/refunds

POST /cinematic/{company_id}/payments/{payment_id}/void

Company API

CRUD - managed by super-managers only, managers can access their own data

GET /cinematic/company/

GET /cinematic/company/{company_id}

POST /cinematic/company/

PUT /cinematic/company/{company_id}

DELETE /cinematic/company/{company_id}

Stores API

CRUD - managed by managers only, accessible by employees and users

GET /cinematic/{company_id}/stores

GET /cinematic/{company_id}/stores/{store_id}

POST /cinematic/{company_id}/stores

PUT /cinematic/{company_id}/stores/{store_id}

DELETE /cinematic/{company_id}/stores/{store_id}

4. Operational environment

4.1. Geography

The PoS system is primarily designed for Lithuanian and other Baltic countries. However, it should be adaptable for expansion to other European Union countries in the future.

4.2. Businesses

System components are customized for businesses in the bar, cafe, restaurant, and Beauty/Spa sectors.

4.3. Usage patterns

We anticipate that businesses may commence operations around 7:00 AM and conclude by 6:00 PM. However, bars typically open later and may operate into the early morning hours until 5:00 AM. This leaves a two-hour window for maintenance activities in different regions, considering that European countries span from UTC+0 to UTC+3 time zones, and Baltic countries specifically observe UTC+2.

Peak hours are expected on Fridays and Saturdays in the evenings, with holidays contributing to heightened activity in bars. Cafes and restaurants, on the other hand, experience the highest customer traffic during lunch hours. Beauty and Spa establishments might see an increased influx during the evenings. Consequently, it can be infered that there will be a rise in usage during both day and evening hours, with a substantial surge in activity during holiday periods.

4.4. User load

It is imporant to mention that a rough estimate will be provided. According to the research report "Representativeness of the European social partner: Hotels, restaurants and café (HORECA) sector" by Eurofound pusblished in 2016 (the number of bussinesses has grown, but the pandemic has prevented significant idustry growth) the number of companies in the HORECA sector was 3,556 in Latvia, 5,519 in Lithuania and 2,695 in Estonia. Which makes for a total of 11,770 companies.

It is expected that about 1% of such a market size is to use this PoS system in the first year. Which is about 118 enterprises.

An enterprise on average might have 100 concurrent users. (It is influenced by business size and operations, number of service points, staffing levels, with each bussiness having different characteristcs. A small cafe might have 1 to 5 users while a large restaurant establishment might reach numbers into the hundreds).

Thus it can be estimated a load of about **11,800 users**. Though the number is expected to be higher during the holidays or other special events.

4.5. Write-Read ratio

Users interacting with the system, will predominantly perform read operations for tasks such as checking for available services or items, orders and appointment management, as well as their statuses and transactions. Meanwhile write operations - crucial for maintaining transactional data and adhering to security and data integrity - are expected to be less frequent, although still fairly common, as it would likely be event-driven, occurring during specific actions like customer transactions, order or inventory status updates, new product or appointment entities being introduced. Hence, the PoS system anticipates a write-read ratio skewed towards read operations, approximating to **30:70**. This aligns with the emphasis on efficient query performance.

4.6. Internalization

The system spans various user interfaces, which might require response representation in different lengths, so the idea for the API is to present its responses only in English and have the consuming application responsible for the translations to the language necessary. The dates would be represented as a ISO Date string and currencies depicted in Euros (as the system is designed in the context of the Baltics, with a possible expansion to a broader EU clientele).

4.7. Law

The European Union retail environment is highly regulated. General Data Protection Regulation (GDPR), Payment Card Industry Data Security Standard (PCI DSS) must be considered.

4.8. Sensitive data

From the definition of a PoS system, the assets to be protected are the audit trail (for accountability, security, compliance with regulation, data loss prevention and etc), customer payment information, PII and other sensitive personal data. The data, not necessary for business needs should not be collected.

4.9. Integration and dependencies with other systems

The PoS system will rely on external payment gateways for transaction processing.

5. Non-functional requirements

The non-functional requirements (further referred to as **NFR**) section focuses on performance, scalability, legal and security requirements. They were elicited keeping the operational environment in mind.
The UX and infrastructurea are not the focus of this specification. This is due to the fact that the system provides APIs and does not have a UI, hence the UX NFRs are limited to aspects that impact users interacting with the system indirectly through the APIs. Furthermore, there is not a lot of context in terms of infrastructure like the hardware or network the POS system will run on.

5.1. Performance

1. Scalability
   NFR-1-1 The system's logical architecture should be able to sustain at least a 1% increase in transactional load on a yearly basis.
   NFR-1-2 The system should easily adapt to new requirements imposed by changes in legislation.
   NFR-1-3 The system must support the addition of new locations to the network with centralized data management, without impacting performance at existing locations.
   NFR-1-4 The system should exhibit horizontal scalability to accommodate fluctuations in user demand, particularly during peak hours and periods.
2. Capacity
   NFR-2-1 The system should be able to effectively serve simultaneously up to 12,000 active users.
   NFR-2-2 Response time of the system should not exceed one second for the execution of 90% of simple queries for at least 12,000 concurrent active users during normal working hours.
   NFR-2-3 Response time of the system should not exceed three seconds for the execution of 90% of complex queries for at least 12,000 concurrent active users during normal working hours.
3. Availability
   NFR-3-1 The system shall not have any single point of failure (redundant components, load balancing, support for failover).
   NFR-3-2 Unless the system is non-operational, the system shall present the user with a notification informing them that the system is unavailable.
   NFR-3-3 System up-time (software, hardware, network) of at least 91% (or less than 730 hours of unavailability per year).

5.2. Legal requirements

4. Security

   NFR-4-1 Passwords for secure logins must be stored encrypted.

   NFR-4-2 The User Management module should identify the different users accessing the system in a secure and traceable way.

   NFR-4-3 The system shall guarantee that authenticated users can only access services or data matching their role and access rights.

   NFR-4-4 Information about actions performed on records, the time of these actions(view, search, creation, edit, and deletion), and the user who performed the action must be securely stored.

   NFR-4-5 Ensure compliance with PCI DSS for handling transactions and secure storage of customer payment details.

5. Data integrity

   NFR-5-1 Whenever a change is made to information stored in database, the fact of the change shall be recorded in a database or equivalent technology that is routinely backed up. This is intended to identify changed information in the event of the loss of data.

   NFR-5-2 The system should allow administrators to set up a functionality for archiving, restore data, creation of backup copies, and scheduled maintenance.

   NFR-5-3 The system should register all system events and errors, status of exchanged messages, etc.

   NFR-5-4 The system log should contain the following data: date, time, system process, type/nature of actions, system error message.

6. Regulatory

   NFR-6-1 Before performing any processing of personal data, the relevant legal basis according to Art. 6 GDPR shall be identified and documented and consent by the user, who is presented with the documentation, for processing must be given.

   NFR-6-2 The system may only collect and process data that are necessary for the defined and documented purpose. This includes each individual data attribute as well as the overall data set.

   NFR-6-3 Data processed and stored shall be accurate and, where relevant, kept up to date.

   NFR-6-4 The system shall provide the controller with the ability to identify any personal data no longer required.

   NFR-6-5 The system should provide the controller with the ability to delete personal data identified as no longer required, including all instances of the data such as backups.

   NFR-6-6 Maintain an audit log of all transactions and administrative actions to comply with the EU's financial record-keeping requirements.

5.3. Usage context

7. Interoperability

   NFR-7-1 The implementation of the system should follow open standards and use well-known and widely accepted technologies in order to ensure interoperability.

   NFR-7-2 Consistent attribute naming should be used to deliver data in mutually understood way between all actors in the system.

5.4. User experience

8. Usability

   NFR-8-1 The system shall provide clear and informative error messages and status codes in API responses.

   NFR-8-2 API is to present its responses in English.

   NFR-8-2 Dates represented as a ISO Date string.

   NFR-8-3 Currencies depicted in Euros.

5.5. Infrastructure, deployment

9. Reliability

   NFR-9-1 The system shall roll back all updates when any update fails to commit.

10. Recoverability

    NFR-10-1 The system must be able to recover any transaction data from backup systems with no more than 30 minutes of data loss in the event of a failure.

    NFR-10-2 Implement a point-in-time recovery system that ensures transaction data can be recovered to any moment within the last 24 hours.

6. Scaling

Scaling adheres to the specified NFRs outlined in Section 5 of the document.

6.1. Regional sharding

Regional sharding is a database partitioning strategy that concentrates on the distribution of data according to geographical locations, with a current focus on Lithuania, Latvia, and Estonia, and a future expansion plan to include other European Union countries. This approach facilitates the accommodation of increased transactional load (NFR-1-1) by dispersing both data and processing loads across multiple regions. Furthermore it addresses new location integration (NFR-1-3) and legislative adaptability (NFR-1-2) as regional sharding will enable centralized data management, allowing new locations to be integrated more seamlessly into the system and aid in quicker adherence to local laws.

6.2. Sharding by client

In case the regional database becomes a bottleneck (e.g 6.4 a lot of instances are created and there is a high volume of requests), it should further be sharded by clients.

6.3. DNS load balancing

DNS load balancing should be utilized to route traffic to the closest region, optimizing response times by directing users to the server with the lowest latency based on their geographical location.

6.4. Horizontally scalable server instances

Horizontal Scalability for Peak Demand (NFR-1-4): to ensure the system can seamlessly handle increased user demand, especially during peak hours, a strategy of dynamically scaling server instances horizontally will be employed in each region as needed. During peak hours, as the volume of incoming traffic surges, additional server instances (pods) will automatically launch to efficiently distribute the increased workload.

6.5. Internal load balancing

Will distribute the load across different servers within the region.

7. System design changes from previous document

- Loyalty information was missing in the previous team's system design. Purchases would now add points to the user account, which could then be used to redeem various offers created by a manager through the loyalty related endpoints in the User API. User would include an additional loyaltyPoints property.
- Previous team considered an empty order to be non-existent. This has been found to be against the common processes, hence it was decided to allow such workflow.
- In previous team's 6.1.1 Account management they highlight that a user can be assigned both *UserRole*s (which consist of user permissions) and *UserPermission*s itself. This introduces significant ambiguity, as there is no definite way to deal with permissions over some of the workflows, where these two may clash (e.g. setting a role, then trying to remove a specific permission granted by the role or setting a permission, adding a role with that permission, removing the role).
- Service options were removed, as we collectively didn't find a proper workflow or situation, where it would be necessary. If for example we had a barbershop, it may have a service created for both hair trimming and beard trimming, and the client would take two consecutive appointments.
- There was no information for how companies or stores are created, but these entities were depicted in the data models. A seperate company API, accessible only by a previously undefined "super-manager" role, which would be the whole PoS administrators, as well as stores API have been created.
- Services have been missing taxes property, which has been added.