Software Requirements Specification

for

CONNECT

Version 1.3

Prepared by: Dávid Csirák, Péter Balázs Tóth and József Márk Varga

University of Debrecen, Faculty of Informatics

2023.11.21

Table of Contents

[1. Introduction 1](#_Toc151487881)

[1.1 Purpose 1](#_Toc151487882)

[1.2 Intended Audience and Reading Suggestions 1](#_Toc151487883)

[1.3 Product Scope 1](#_Toc151487884)

[1.4 References 1](#_Toc151487885)

[2. Overall Description 2](#_Toc151487886)

[2.1 Product Perspective 2](#_Toc151487887)

[2.2 Product Functions 2](#_Toc151487888)

[2.2.1 Ticketing module 2](#_Toc151487889)

[2.2.2 Live feed module 2](#_Toc151487890)

[2.2.3 Event module 2](#_Toc151487891)

[2.2.4 User module 3](#_Toc151487892)

[2.2.5 Recommender module 3](#_Toc151487893)

[2.2.6 Notification module 3](#_Toc151487894)

[2.3 User Classes and Characteristics 3](#_Toc151487895)

[2.4 Operating Environment 3](#_Toc151487896)

[2.5 Design and Implementation Constraints 4](#_Toc151487897)

[2.5.1 Ticketing module 4](#_Toc151487898)

[2.5.2 Live feed module 4](#_Toc151487899)

[2.5.3 Recommender module 4](#_Toc151487900)

[2.5.4 Notification module 4](#_Toc151487901)

[3. External Interface Requirements 4](#_Toc151487902)

[3.1 User Interfaces 4](#_Toc151487903)

[3.2 Communications Interfaces 4](#_Toc151487904)

[4. System Features 5](#_Toc151487905)

[4.1 Event service 5](#_Toc151487906)

[4.1.1 Description and Priority 5](#_Toc151487907)

[4.1.2 Functional Requirements 5](#_Toc151487908)

[4.2 User service 6](#_Toc151487909)

[4.2.1 Description and Priority 6](#_Toc151487910)

[4.2.2 Functional Requirements 6](#_Toc151487911)

[4.3 Ticketing module 6](#_Toc151487912)

[4.3.1 Description and Priority 6](#_Toc151487913)

[4.3.2 Functional requirements 7](#_Toc151487914)

[4.4 Live feed module 7](#_Toc151487915)

[4.4.1 Description and Priority 7](#_Toc151487916)

[4.4.2 Functional requirements 8](#_Toc151487917)

[4.5 Recommender module 8](#_Toc151487918)

[4.5.1 Description and priority 8](#_Toc151487919)

[4.5.2 Functional Requirements 8](#_Toc151487920)

[4.6 Notification module 9](#_Toc151487921)

[4.6.1 Description and priority 9](#_Toc151487922)

[4.6.2 Stimulus/Response Sequences 9](#_Toc151487923)

[4.6.3 Functional Requirements 9](#_Toc151487924)

[5. Other Nonfunctional Requirements 10](#_Toc151487925)

[5.1 Performance Requirements 10](#_Toc151487926)

[5.1.1 Ticketing module 10](#_Toc151487927)

[5.1.2 Live feed module 10](#_Toc151487928)

[5.1.3 Recommender module 10](#_Toc151487929)

[5.1.4 Notification module 10](#_Toc151487930)

[5.2 Security Requirements 10](#_Toc151487931)

[5.2.1 Ticketing module 10](#_Toc151487932)

[5.2.2 Live feed module 10](#_Toc151487933)

[5.2.3 Recommender module 10](#_Toc151487934)

[5.2.4 Notification module 11](#_Toc151487935)

[5.3 Bottlenecks 11](#_Toc151487936)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Fetching ideas, defining functionalities | 2023.09.27 | - | - |
| Organizing services | 2023.10.24 | Simplifying the original idea(s) into services. | 1.0 |
| Describing common parts of the application | 2023.11.07 | Filling common parts in the SRS document. | 1.1 |
| Merging service descriptions and diagrams | 2023.11.08 | Completing service-based descriptions in the SRS document. | 1.2 |
| Finishing SRS and the presentation. | 2023.11.21 | - | 1.3 |

# Introduction

## Purpose

This is the documentation of the 1.0 version of the event manager application named **CONNECT**. The document covers and describes the entire system.

## Intended Audience and Reading Suggestions

This document is primarily intended for developers, testers, and project managers. Users can also gain a lot by studying the documentation; however, certain details might be too technical and therefore should be omitted by them.

The rest of this SRS contains all the necessary and relevant information required to plan, develop, and test the product to be deployed at the end of the software development lifecycle. The second chapter contains the overall description of the software including the motivation behind the application, its functionalities, expected user characteristics, limiting factors and other related subjects.

In the subsequent chapters, we will summarize the functional and non-functional requirements of the product, as well as the most important interface requirements that are present in the software.

The order in which you wish to go through the chapters really depends on the group of readers you belong to. Regardless of the aforementioned reader group, starting with the first introductory chapter is advisable since it gives the reader an idea about the whole project. If you are a tester, you might prioritize the functional specification and leave the less relevant information for a later time. Potential users would probably prefer studying the overall description of the application, since it describes the behavior of the product and other data that the user might be interested in.

## Product Scope

The subject of the documentation is the application named Connect. Its main purpose lies in the endeavor which strives to eliminate the biggest hindrances in the utilization of ordinary event manager applications, namely their functional deficiencies and other bad features. This application intends to provide an interface for event management for both organizers and participants and aims to make event organization, attendee registration, and other event-related tasks an easy, fun, and transparent process. Apart from the core functionalities that are indispensable for the proper working of the product, the project wishes to provide the users with additional features which enhance the user experience and foster interactivity between organizers and participants.

## References

* **Amazon AWS S3** – Cloud Object Storage
  + <https://aws.amazon.com/s3/>
* **Chat API**: Talk JS
  + <https://talkjs.com/>
* **Bank API**: Stripe
  + <https://stripe.com/en-hu>
* **Recommender Engine:** Recombee <https://www.recombee.com/>

# Overall Description

## Product Perspective

The main idea of developing this product was the lack of flexibility of the other event manager competitors, like Facebook events and others. Mostly, the problems were, the badness of the notification system, the poor functionality list etc.

It is known that there are plenty of event management systems/applications, but none of them have this wide of a function palette, which will be deeply described in the following chapters of the documentation.

## Product Functions

### Ticketing module

* Users can purchase and then manage their tickets.
  + their purchases will be checked before checkout due to free space and age restrictions
* There will be almost no delay in purchasing the tickets, because of the system design.
  + all the requests will go simultaneously
* The invoice will be generated automatically, so that the user can download the respective invoice after the transaction completed
  + it invokes a premade invoice generator for easing the generation process
* Users can export their tickets along with the optional invitations.
  + these will be NFT’s for design and authenticity purposes

### Live feed module

* There will be live stories, forum, comments, videos, pictures relating to the actual event.
  + comment and image stream almost like on Twitter
* There will be group chats for forming new social/friend groups.
  + group chat types will be: **event-type based**or **event based**chat rooms
* There would be some QR codes on the event site, where people can take group photos or videos and then share it to the common group/event feed of the event.
  + there would be some limitations like the maximum number of uploaded pictures per user (for example 5 images) for handling unexpected data growth
    - this also ensures that the text-image ratio is balanced on the live feed of the event

### Event module

* Authorized users can manage their events.
  + create events, and updating, viewing and deleting them afterwards
* Users can create 2 types of events.
  + ‘Organization’ and ‘user’ events
* There are 2 possible user responses to a given event.
  + ‘Going’ and ‘Interested’
* When creating events, organizers are able to define registration constraints.
  + age restriction, maximum capacity
* There is an option to specify the performers at an event.
* Users can rate events.

### User module

* Provides registration and login features.
* There is the option of modifying one’s personal data.
* Users can sign up for events in solo or in groups after authentication.
* Provides authorization features, which allows e.g., organizers to manage their events.
* Users can register and log in via Google-, Facebook- or other accounts as well.

### Recommender module

* There will be personalized event recommendations based on user’s location, calendar, participated events, likes, invitations and overall activity.
* Upon signing up, there will be a form where the users can fill in their interests in the types of events, areas, pricing, starting time, dress code.
* The users will be notified about any new or upcoming event that matches their needs.

### Notification module

* The users will be notified about new events, upcoming events, new posts, changes made to the events, canceled events and more
* The notifications will be sent via
  + Email
  + SMS
  + Push

## User Classes and Characteristics

In case of this event manager application, there are two classes of users with different privileges and abilities.

The first group is represented by those who organize different events – the organizers. They possess higher privileges than members of the other user group, since they can create, managing and deleting their own events, and making any event-specific information available to the public. Their capabilities represent a superset of the regular users’ privileges, meaning that they can do everything that someone who is seen as a participant by the system can do.

The other group is represented by the participants of the events. Members of this class are in possession of lower privileges than organizers. Consequently, they are not able to create and handle events. However, they can utilize all the helpful features that make the product easy and fun to use. These skills include expressing their interest in the event by marking it with the ‘Going’ or ‘Interested’ tags, rating events, using the live feed of each event to stay up to date with the ongoing programs on site, buying tickets and much more.

## Operating Environment

The operating environment of the system could be any arbitrary server or servers. It needs to be able to work together at least with the operating system and another program that can allocate resources for it can control its scaling depending on usage.

## Design and Implementation Constraints

### Ticketing module

The limitation related to the ticketing module would be the huge purchase of the tickets, when everybody or a relatively huge user base tries to purchase tickets for the event. It can result system slowness, or it could also produce errors or lags in the application or event the failure of the ticket purchase.

Also, there could be security issues relating to the bank information. Precisely, when the data is sent to the server to be processed by a third-party transaction API and stored in the database. Speaking of the database, the payment information of a user, like card holder name, CVC etc. should be encoded inside the database. Continuing, authentication and authorization procedures should be used, like JWT, OAuth2 etc.

### Live feed module

This module does not really have any implementation constraints, because all the sub-modules are strictly separated from each other, so the communication between the client and the server can be asynchronous.

### Recommender module

The implementation constrain of the recommendations could be finding a good enough multi-class classification model, which helps determining the specialized event recommendation based on user’s interests.

### Notification module

This module’s design related constrain could be, that every part of the application needs to be able to send their own notification in an asynchronous mode. This is a crucial point.

# External Interface Requirements

## User Interfaces

When developing the user interface, multiple design principles would be taken into consideration, like adaptiveness, responsiveness, and other important factors. These steps are required to use the application optimally and fluently across all the devices, let it be a personal computer, a phone, or a tablet.

Also, the simplicity of the GUI will be important too, for easing the usage and to increase the user experience.

For these tasks, a premade styling library will be used, which in our case is Tailwind or Material UI. These components allow using all the principles of the points mentioned above.

## Communications Interfaces

Mainly, the application uses **HTTP** or **HTTPs** protocols when communicating across/with the services, but in some cases, like the *Ticketing module* or even the*Live feed module* **WebSocket** protocol will be used for providing a simultaneous two-way communication between the client and the server.

# System Features

The main diagram representing the whole application can be found on **Figure 6,** in the Appendix B: Analysis Models section.

## Event service

See on **Figure 4.**

### Description and Priority

This feature provides the basic functionalities for interacting with events including creating, updating, and deleting events, and viewing the event details. The listed options are only available for logged in and authenticated users. It also provides the ability to create different types of events, communicating with other participants, and other useful features.

Since some of these features are of uttermost significance, it gets a priority rating of High.

### Functional Requirements

REQ-1: Logged in/Authorized users should be able to create events.

REQ-2: Logged in/Authorized users should be able to delete their own events.

REQ-3: Logged in/Authorized users should be able to view event details.

REQ-4: Logged in/Authorized users should be able to update their own events.

REQ-5: Users can set two states for a particular event: ‘Going’ or ‘Interested’.

REQ-6: Users should be able to create ‘organization events’ if they are registered as an organization or institute, e.g., University of Debrecen

REQ-7: Users can create ‘user events’, which are smaller events, e.g., organized by a group of friends, although the number of attendees is not set in stone.

REQ-8: Users should be able to create age-restricted events, where people not meeting the age requirements cannot register and participate in the event.

REQ-9: Users can rate each event – upvote or downvote.

REQ-10: Every event should have a separate page where registered users and organizers can communicate with each other.

REQ-11: Organizers should have the ability to set the maximum number of participants for their events.

REQ-12: Organizers should be able to set the list of performers for the event.

## User service

See on **Figure 3.**

### Description and Priority

The user service is probably the most fundamental part of the entire application, since it provides all the necessary tools for managing users.

This feature is responsible for the registration process as well as logging in. Authentication is also one of the tasks of this service which makes sure that only registered users can sign up for a particular event. The user service is also in charge of handling authorization, which comes in handy, for example, when having to make a distinction between organizers and regular participants.

Due to the nature of the abovementioned features, this service is of uttermost significance and therefore receives a ‘High’ priority level.

### Functional Requirements

REQ-1: Everyone should be able to register and create their own account.

REQ-2: Every user must have the ability to log in after they have registered.

REQ-3: The user service must support several methods of logging in, including the registered username and password, and other possibilities, e.g., Google of Facebook account.

REQ-4: Users can fill out their profile data and modifying them at any time.

REQ-5: Users must be able to register for an event either in solo or in groups, with other users’ consent.

REQ-6: Signing up for an event (REQ-5) must be preceded by user authentication, which can happen via email, or Google-, Facebook- or other accounts.

REQ-7: After a user has created an event, the organizer role will be assigned to him/her, which enables the person to manage that particular event according to his/her needs.

REQ-8: An additional admin role should be assignable to the administrators, so they are capable of handling and modifying everything in the entire application. This role must not be made available for users neither on the registration page nor in any other places in the software.

## Ticketing module

See on **Figure 1.**

### Description and Priority

This feature of the application has a very high priority, since users cannot attend events without their tickets purchased. The cost of this module would be almost the highest one too, because a fast and simultaneous communication and process handling is required.

Also, this module enables users to purchase their tickets relatively fast, to download their invoices, to securely reuse the payment data submitted by them in other transactions, and it also lets users to generate their own NFT ticket for the event, and to generate invitations too for other persons.

### Functional requirements

REQ-1: The user can purchase tickets.

REQ-2: The user has the rights to refund a ticket if it is no longer needed.

REQ-3: The user is not allowed to purchase a ticket if there are no free tickets left for the event, or if she/he is under the required age of the event.

REQ-4: The ticket purchase response cannot take too long.

REQ-5: The ticket purchase should be done in a safe environment and not leak any sensitive information.

REQ-6: The applications availability status should always be above 99 percent.

REQ-7: All the communication across the sub-modules should be done using asynchronous communication methods.

REQ-8: If the transaction succeeds, then the invoice must be generated and sent back to the client.

REQ-9: If the user wants to save her/his payment information, it is supported and safely saved in the database.

REQ-10: The communication between sensitive sub-modules (like sending payment data to the server or to the third-party bank API) should always be secure.

REQ-11: The ticket is automatically generated along with the NFT part and then they are sent back to the client.

REQ-12: The design of the NFT must be similar to the respective event’s design.

REQ-13: Sensitive information is always encrypted when storing in the database.

REQ-14: A QR code is also generated to verify the tickets in-person.

REQ-15: The user is always notified about every significant procedure, operation.

## Live feed module

See on **Figure 2.**

### Description and Priority

This module is responsible for managing group chats for each event, which consists of handling files and texts (chat, comments, and photos). These processes can be done asynchronously because they are not related to each other. The only thing that is common in all of them, on the database side, is the event’s identifier.

Regarding the priority, the module and its sub-modules has a reduced, medium priority, due to the fact, that this feature is just one example of an extra feature for building up the community of a certain event or even in general.

### Functional requirements

REQ-1: Authenticated users can place comments on events.

REQ-2: Authenticated users can use the chat functionality of the application.

REQ-3: Users cannot comment cursing words to the comment section or to the chat.

REQ-4: Users can post - a maximum number of 5 - pictures to the feed, which is then saved in the database, which then can be accessed, for generating memories from an event.

REQ-5: The response latency of each sub-module cannot be too large.

## Recommender module

See on **Figure 5.**

### Description and priority

This module is responsible for tagging the contents and events in the application. Based on the tags and by analyzing past event participation, geographical preferences, and overall user activity, this module ensures that every recommendation is a tailored invitation, inviting users to explore events that align with their interests and enhance their social calendar.

As the user base expands, the recommender module should scale efficiently to continue delivering timely and relevant event suggestions to all users, ensuring a consistently high-quality experience.

The module should be adaptive, learning and evolving alongside user preferences and changing trends to maintain its effectiveness in providing valuable event recommendations over time.

### Functional Requirements

REQ-1: The module should continuously gather and analyze data on events, including types, themes, locations, and user participation history.

REQ-2: Implement machine learning algorithms (or outsource it) to process user data and event information, generating personalized recommendations based on user behavior, preferences, and location.

REQ-3: The system must generate recommendations in real-time, considering the most recent user activity and event updates to provide up-to-the-minute suggestions. It will be handy to consider a database with caching ability.

REQ-4: Incorporate a feedback mechanism that allows users to provide explicit feedback on recommended events, enabling the system to continuously learn and refine its recommendations.

## Notification module

See on **Figure 5.**

### Description and priority

This module serves as the communication bridge between the application and its users, delivering timely and relevant updates to enhance user engagement. This module is responsible for conveying personalized event invitations, reminders, and important information, ensuring that users stay informed.

### Stimulus/Response Sequences

### Functional Requirements

REQ-1: Registered users can receive notifications.

REQ-2: Users interested in a given event should receive any notification about the event.

REQ-3: The system should allow users to set and customize their notification preferences, specifying the types of events or updates for which they wish to receive notifications.

REQ-4: Notifications must be delivered in real-time to ensure that users receive timely updates on recommended events, reminders, or any other relevant information.

REQ-5: The system should support notification delivery across multiple platforms, including mobile devices and web applications, providing a consistent user experience.

REQ-6: Implement a mechanism for prioritizing and grouping notifications based on their importance and relevance, allowing users to easily manage and prioritize their notifications.

# Other Nonfunctional Requirements

## Performance Requirements

### Ticketing module

This module will require a powerful processor for handling all the asynchronous processes like purchasing tickets. On the other hand, a fast hard drive for managing database operations would be an advantage too.

### Live feed module

The only important part of the hardware requirement for this module would be the speed of the hard drive for seamlessly saving and retrieving information of a feed (texts and files). On the other hand, a faster processor would also fasten the procedures which are required in order to use this module fluently.

### Recommender module

* The recommender module must be capable of scaling to accommodate a growing user base and increasing volumes of event data without a significant degradation in performance.
* The machine learning algorithms powering the recommender module should be optimized to execute efficiently.
* A caching mechanism should be implemented to store and retrieve frequently accessed recommendation data.

### Notification module

A performance requirement is needed in the delivery time of the notifications. Notifications should be delivered to users within a maximum timeframe of X seconds from the occurrence of the triggering event.

## Security Requirements

### Ticketing module

This module requires a high level of security because it sends and saves payment information in a database, so there is a huge need for security. The data should always be encrypted and safely delivered from one sub-module/service to the other.

### Live feed module

Regarding this module, the only security aspect comes up when storing sensitive images, due to GDPR regulations. Thus, files and sensitive images should be encrypted too when sending them back and forth between the sub-modules or services.

### Recommender module

It is required implementing end-to-end encryption for user data to protect sensitive information such as past event participation and preferences.

A secure communication channel (HTTPs) is needed between the recommender module and other parts of the application or other third-party API-s

### Notification module

It is required that every notification should be delivered to the right user. It is important not to leak sensitive personal information to the wrong user, via notifications.

## Bottlenecks

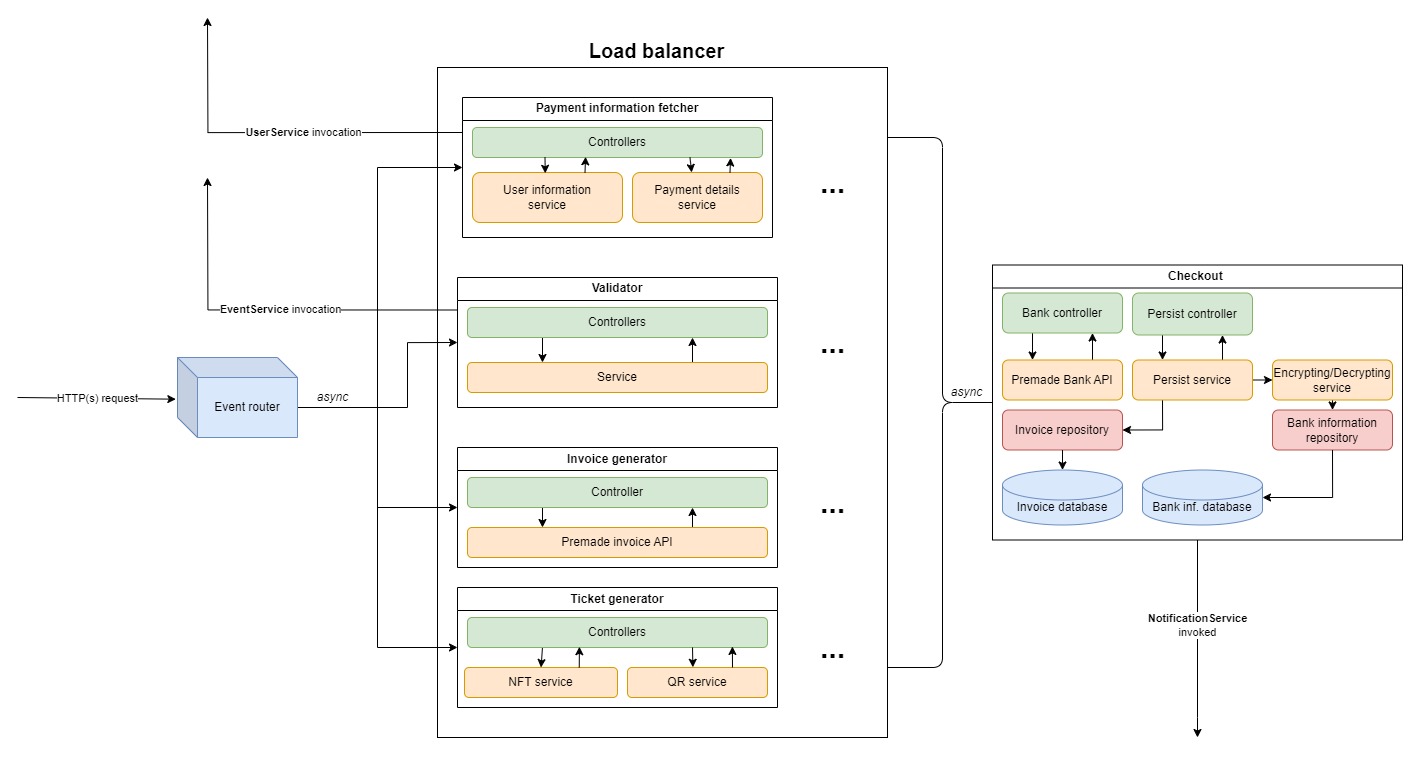
1. The data on the live feed must be up to date all the time.
   1. Using a NoSQL database would eventually fix this problem.
2. There could be huge data operations when using the application.
   1. This problem was solved using load balancing.
3. Handling large user base.
   1. The application was designed to withstand approximately 10.000 users, so multiple database replicas would solve this.
4. Scalability issues.
   1. Here too, the load balancer would solve these issues.
5. Data loss.
   1. The master-slave database concept would prevent this issue form happening.

Appendix A: Glossary

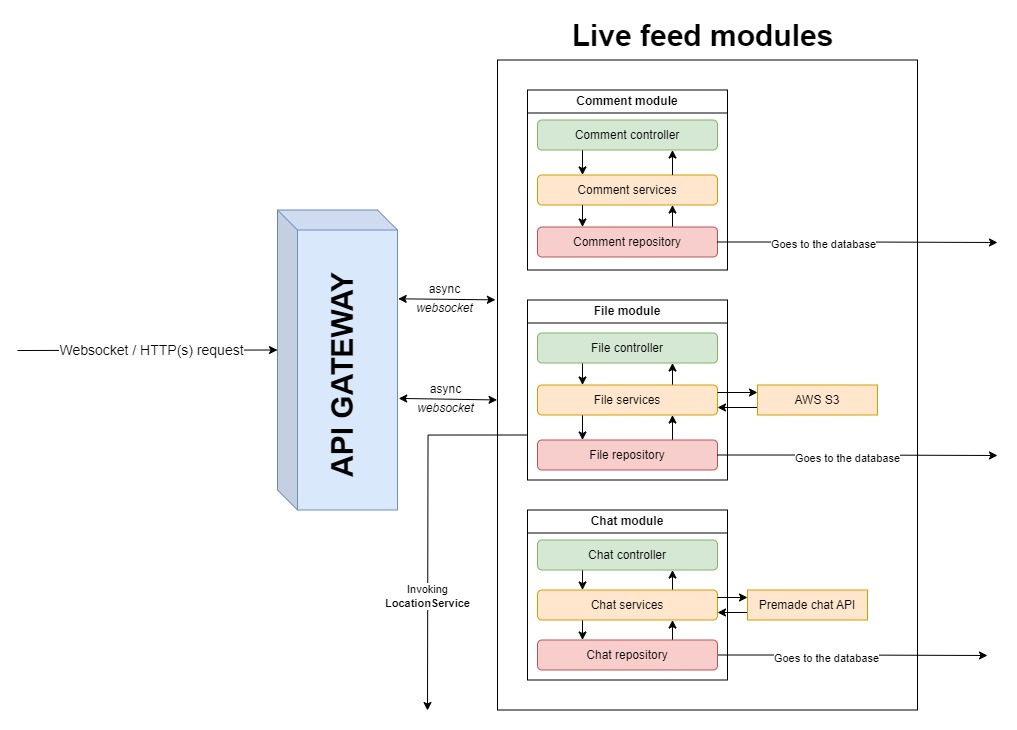
* **NFT:** Non-Fungible Token
* **JWT:** Json Web Tokens
* **OAuth2:** Open Authorization 2
* **QR:** Quick Response code
* **GDPR:** General Data Protection Regulation
* **CVC:** Card Verification Code

Appendix B: Analysis Models

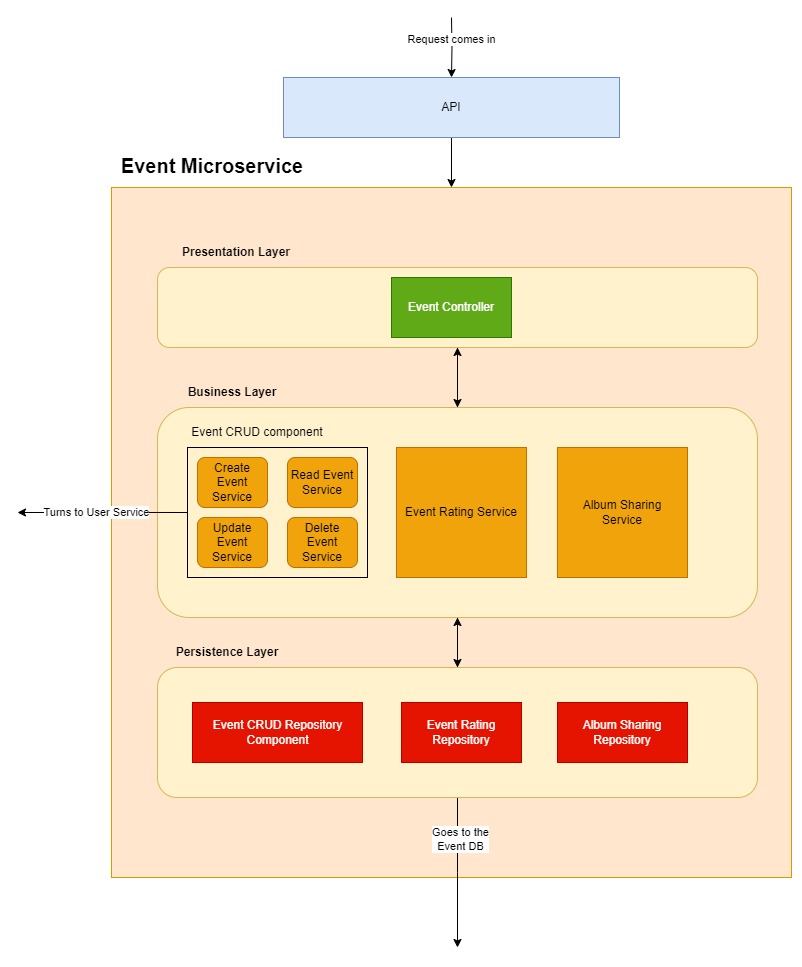
* Figure 1: the architectural diagram of the *Ticketing Module.*



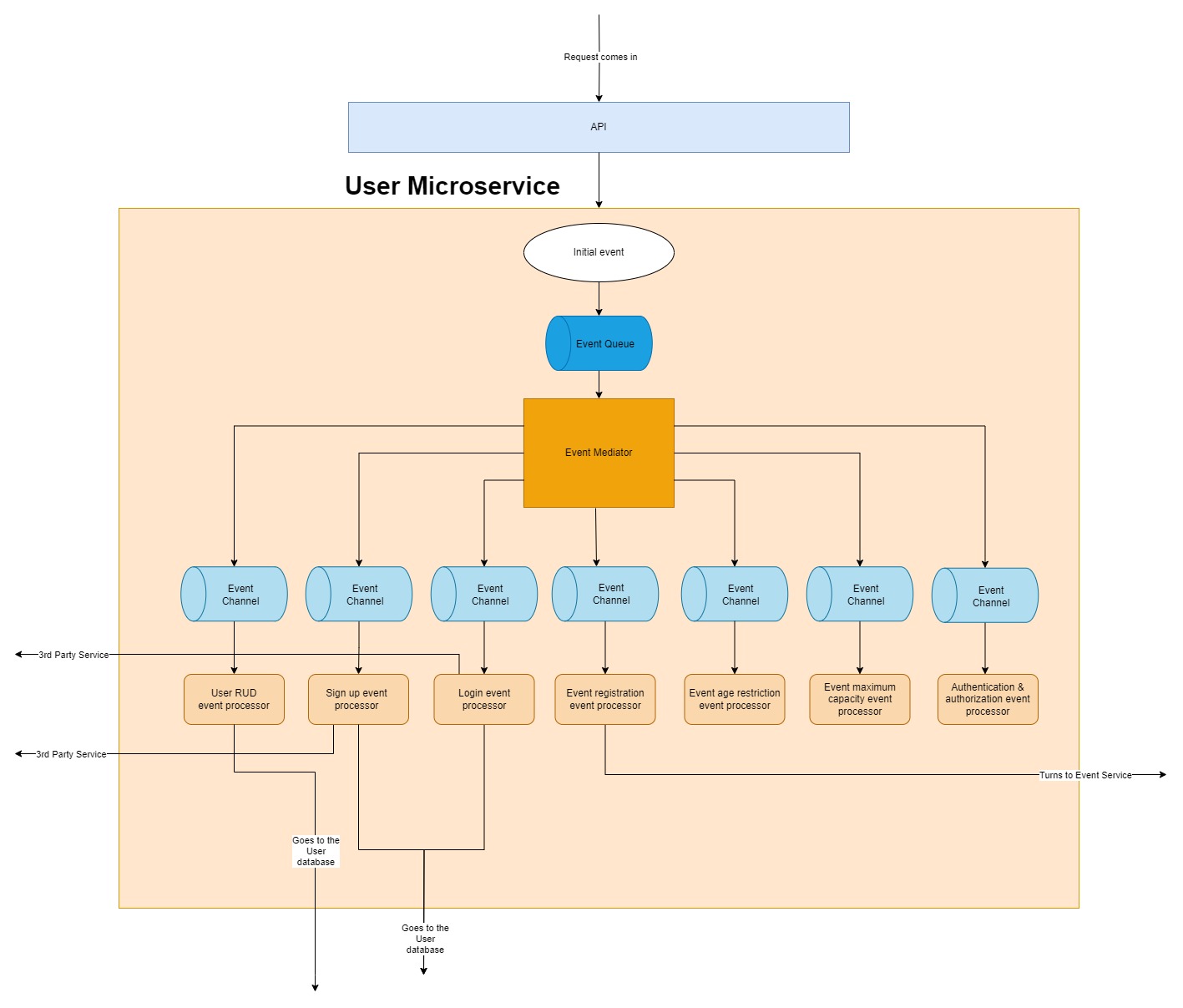
* Figure 2: the architectural diagram of the *Live Feed Module.*



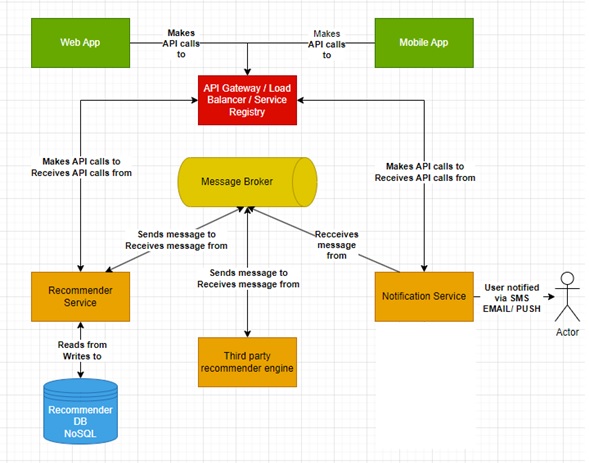
* **Figure 3:** the architectural diagram of the *User Service.*

****

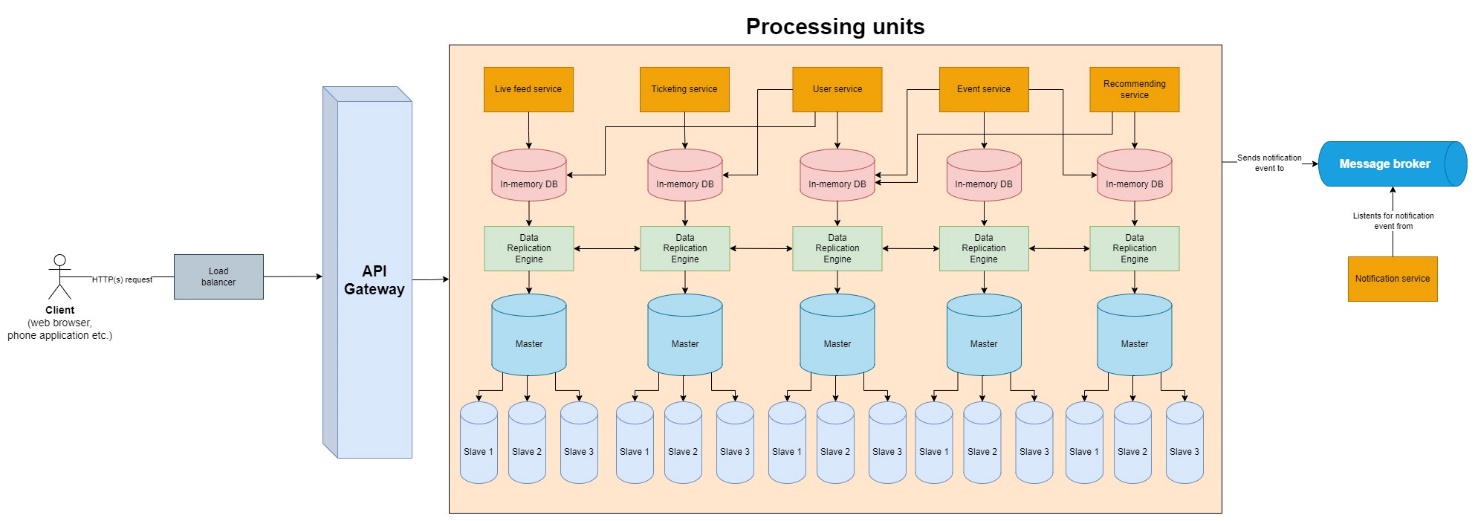
* **Figure 4:** the architectural diagram of the *Event Service.*

****

* **Figure 5:** the architectural diagram(s) of the *Recommender module* and the *Notification module.*



* **Figure 6:** the main architectural diagram of the application.

****