

Trading Cards Application

Name:

Group:

Table of Contents

[Deliverable 1 2](#_Toc2055786399)

[Project Specification 3](#_Toc825512728)

[Functional Requirements 3](#_Toc362225657)

[Use Case Model 1 3](#_Toc324720183)

[Use Cases Identification 3](#_Toc159950011)

[UML Use Case Diagrams 3](#_Toc1353450407)

[Supplementary Specification 3](#_Toc501087280)

[Non-functional Requirements 3](#_Toc1600632703)

[Design Constraints 3](#_Toc729746755)

[Glossary 3](#_Toc178374120)

[Deliverable 2 3](#_Toc1123232848)

[Domain Model 3](#_Toc605147785)

[Architectural Design 3](#_Toc1115440048)

[Conceptual Architecture 3](#_Toc2005513277)

[Package Design 3](#_Toc596515525)

[Component and Deployment Diagram 4](#_Toc1771682286)

[Deliverable 3 4](#_Toc174956409)

[Design Model 4](#_Toc2136237097)

[Dynamic Behavior 4](#_Toc853475638)

[Class Diagram 4](#_Toc2045288274)

[Data Model 4](#_Toc1322833382)

[System Testing 4](#_Toc975007147)

[Future Improvements 4](#_Toc902595900)

[Conclusion 4](#_Toc1374315000)

[Bibliography 4](#_Toc2082771450)

# Deliverable 1

## Project Specification

Scopul acestui proiect este acela de a crea o aplicație web folosind Java Spring Boot și Angular dar și alte instrumente utile precum Postman. Această documentație prezintă aplicația menționată anterior, o aplicație pentru cumpărare, vindere și schimb de cartonașe virtuale.

Pe scurt, în cadrul acestui aplicații, un utilizator va avea posibilitatea fie de a cumpăra un pachet de cartonașe, fie de a cumpăra un cartonaș în mod direct de la un alt utilizator care îl deține deja. Aceste cartonașe au un anumit procentaj care dictează cât de des pot fi obținute dintr-un pachet. Astfel vom avea cartonașe care apar mai des și cartonașe care apar mai rar. Fiecare cartonaș va avea atribuit un jucător de fotbal. De asemenea, vom avea un marketplace în care să poată lista cartonașele cu scopul de a fi cumpărate de alți utilizatori.

## Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Categorie** | **Descriere** | **Nivel de Importanță** |
| 1. | Funcționalitate de bază | Abilitatea unui utilizator de a își crea un cont prin alegerea unui nume de utilizator și a unei parole | Critic |
| 2. | Funcționalitate de bază | Abilitatea unui utilizator de a se autentifica în propriul său cont utilizând credențialele menționate în punctul anterior | Critic |
| 3. | Funcționalitate de bază | Abilitatea de a cumpăra un nou pachet de cartonașe. | Critic |
| 4. | Funcționalitate de bază | Abilitatea de a vizualiza propriile cartonașe acumulate de-a lungul timpului | Critic |
| 5. | Funcționalitate de bază | Abilitatea de a crea o listare pentru un cartonaș în marketplace. | Critic |
| 6. | Funcționalitate de bază | Abilitatea de a cumpăra un cartonaș listat în marketplace. | Critic |
| 7. | U.I. | Aplicația va avea un U.I intuitiv și ușor de folosit care să permită utilizatorului să acceseze toate funcționalitățile importante ale aplicației. | Critic |
| 8. | U.I. | Meniu destinat vizualizării pachetelor disponibile | Critic |
| 9. | U.I | Meniu destinat alegerii unor cartonașe care pot fi afișate în mod public de către un utilizator. | Mic |
| 10. | U.I | Meniu pentru introducerea unor parametrii care să faciliteze găsirea unor anumite cartonașe în marketplace. | Mediu |
| 11. | U.I. | Meniu pentru afișarea jucătorilor din marketplace. | Mediu |
| 12. | Bază de date | O bază de date MySQL descrisă cu ajutorul Java SpringBoot. | Critic |
| 13. | Bază de date | Crearea de relații între tabele. | Critic |
| 14. | Testare | Crearea de teste pentru operațiile de tip CRUD ce se efectuează pentru fiecare tabel în parte. | Critic |
| 15. | Testare | Creare de teste pentru funcționalitățile aplicației. | Critic |

## Use Case Model 1

### Use Cases Identification

Use-Case:

Level:

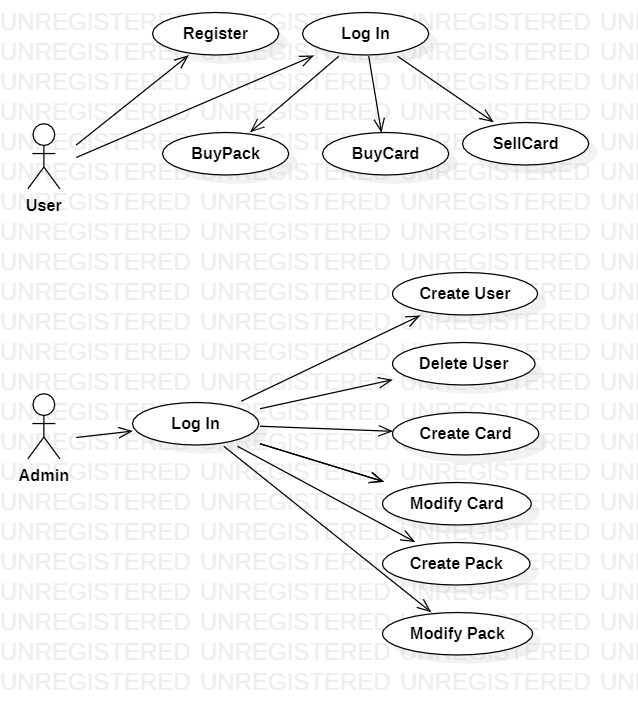
Primary Actor:

Main success scenario:

Extensions:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use-Case** | **Level** | **Primary Actor** | **Main success scenario** | **Extensions** |
| Log In |  | User | Utilizatorul introduce username-ul și parola corecte și este autentificat în cont. | Utilizatorul greșește username-ul sau parola și se afișează un mesaj care îl avertizează de acest fapt. |
| Register |  | User | Utilizatorul introduce un username și o parola și îi este creat un cont nou. | Username-ul există deja sau parola nu respectă anumite specificații afișându-se un mesaj de eroare. |
| Cumpărare pachet |  | User | Utilizatorul cumpără un pachet și primește un anumit număr de cartonașe. | Acesta nu are suficienți bani pentru a cumpăra pachetul și se afișează un mesaj corespunzător. |
| Listare cartonaș |  | User | Utilizatorul creează o listare pentru un cartonaș propriu. |  |
| Cumpărare cartonaș |  | User | Utilizatorul cumpără un cartonaș. | Listarea nu mai există sau a expirat și se afișează un mesaj de eroare. |
| Admin creare User |  | Admin | Administratorul creează un nou user. | User-ul este deja existent și noul user nu poate fi creat. |
| Admin ștergere User |  | Admin | Administratorul șterge un user în funcție de numele acestuia. | User-ul nu există și nu poate fi șters. |
| Creare cartonaș |  | Admin | Administratorul creează un cartonaș nou. |  |
| Modificare cartonaș |  | Admin | Administratorul modifică un cartonaș. |  |
| Creare pachet |  | Admin | Administratorul creează un pachet. |  |
| Modificare pachet |  | Admin | Administratorul modifică un pachet. |  |

### UML Use Case Diagrams



## 

## Supplementary Specification

### Non-functional Requirements

[Choose 4 NF for your system, describe them and explain why these NF are suitable for your implementation. ]

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Categorie** | **Descriere** |
| 1. | Compatibility | Aplicația trebuie să funcționeze în toate browserele moderne. |
| 2. | Compatibility | Aplicația trebuie să funcționeze pentru device-uri low end. |
| 3. | Data Integrity | Aplicația trebuie să se asigure că toate datele care sunt stocate cu acuratețe și consistență. Acest lucru se aplică pentru toate informațiile stocate (Informații despre user, cartonașe, pachete, etc.) |
| 4. | Usability | Aplicația trebuie să fie intuitivă și ușor de folosit. Un utilizator nou trebuie să reușească să înțeleagă cum să se folosească de funcționalitățile aplicație la doar câteva minute după crearea unui nou cont. |

### Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

Aplicația va fi scrisă în Java SpringBoot și Angular. Vor fi folosite principiile SOLID, fiecare clasă având o singură responsabilitate. De asemenea clasele vor putea fi extinse ușor fără a modifica alte funcționalități anterioare. De asemenea va fi respectat principiul încapsulării, astfel vom lăsa descoperite doar acele metode care sunt absolut necesare să fie publice. Restul variabilelor și metodelor ce țin de logica internă a clasei vor fi private. În plus, vom oferi metode de setter și getter pentru acele atribute private care au nevoie de a fi modificate pe parcurs.

Vom folosi design pattern-ul MVC. Astfel aplicația va fi formată din trei mari părți: Model, View și Controller. Model și View nu trebuie să comunice între ele, practic, nu oricare dintre acestea nu va știi de existența celeilalte. Legătura dintre cele două o vom face cu ajutorul parții de Controller . În partea de Model vom avea tot ce ține de bussines logic, tot ce lucrează datele, prelucrează datele sau alte funcționalități ce se folosesc de acestea. În partea de View vom logica ce ține de afișarea datelor obținute în Model. În partea de Controller vom crea legătura dintre View și Model după cum am menționat și mai sus.

## Glossary

[Present the noteworthy terms and their definition, format and validation rules if appropriate.]

# Deliverable 2

## Domain Model

[Define the domain model and create the conceptual class diagrams]

## Architectural Design

### Conceptual Architecture

[Define the system’s conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]

### Package Design

[Create a package diagram]

### Component and Deployment Diagram

[Create the component and deployment diagrams.]

# Deliverable 3

## Design Model

### Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

### Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

## Data Model

[Create the data model for the system.]

# System Testing

[Describe the testing methides and some test cases.]

# Future Improvements

[Present some features that apply to the application scope.]

# Conclusion

# Bibliography