

**O0500 - SOFTWARE ARCHITECTURE**



|  |  |
| --- | --- |
| **Version:** | 1.0 |
| **Status:** | Draft |
| **Approver:** | Ngô Thái Bình  Nguyễn Thị Diễm Trang |
| **Author:** |  |

**Dokumenthistorik**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Author | Status | Bemærkninger |
| 1.0 | 20 - 12 - 2020 | Bui Khanh Hung  Tran Kien Hung  Pham Duc Quoc Khanh  Vo Ngoc Nguyen Minh |  |  |

**Referencer**

|  |  |  |  |
| --- | --- | --- | --- |
| Reference | Title | Author | Version |
| D0160 | D0160 - User Interface Design | Carsten Rostgaard | 1.0 |
| A0100 | A0100 - Clarification Report | Bui Khanh Hung  Tran Kien Hung  Pham Duc Quoc Khanh  Vo Ngoc Nguyen Minh | 1.0 |

Content

[**Introduction**](#_gjdgxs) **7**

[Purpose](#_3pm5hfs3j3af) 8

[Audience](#_qv4d8hrxnrbt) 8

[Definitions and abbreviations](#_4ih59dc9d0xy) 8

[Reading instructions](#_hj4hq57byub1) 8

[Restrictions and outstanding](#_tyjcwt) 8

[**Solution overview**](#_3dy6vkm) **9**

[Technical solution overview](#_1t3h5sf) 9

[Functional solution overview](#_2s8eyo1) 9

[**Architectural principles**](#_26in1rg) **11**

[Modules](#_lnxbz9) 12

[Loose clutch](#_35nkun2) 12

[Cohesion](#_1ksv4uv) 12

[Software](#_44sinio) 12

[Data validation](#_2jxsxqh) 12

[Data security](#_z337ya) 12

[Network security](#_3j2qqm3) 12

[Performance](#_1y810tw) 13

[Availability](#_4i7ojhp) 13

[Robustness](#_2xcytpi) 13

[Error handling](#_1ci93xb) 13

[Data](#_qsh70q) 13

[Persistence](#_3as4poj) 13

[History and logging](#_49x2ik5) 13

[Rules, texts and system parameters](#_2p2csry) 13

[Database access](#_23ckvvd) 13

[Test](#_41mghml) 14

[Unit test](#_2grqrue) 14

[Component test](#_vx1227) 14

[Integration test](#_3fwokq0) 14

[Drift](#_1v1yuxt) 14

[Logging](#_4f1mdlm) 14

[Surveillance](#_2u6wntf) 14

[Modular user interface](#_19c6y18) 14

[Templates](#_3tbugp1) 14

[**Use cases**](#_28h4qwu) **14**

[Login](#_hw1j1vhumk0a) 18

[Login with email](#_d5gcl54texgk) 18

[Login with social credentials](#_v96v57hr6xix) 18

[Logout](#_8411fpt59gnz) 19

[Sign up](#_q937btjn3f0b) 20

[Sign up with Email](#_v96v57hr6xix) 20

[Sign up with social credentials](#_v96v57hr6xix) 20

[Find password if forgot](#_xsjyu44nscyv) 21

[Overview current event](#_yiecohohjv6i) 22

[Send announcement in waiting room](#_vjiy8s1k18cv) 23

[Overview upcoming event](#_p8raopdegzw5) 23

[Create event](#_z902p340fab9) 24

[Search/Filter event](#_w1xeayvxsa39) 25

[Overview question bank](#_hcz3v8dvnptl) 26

[View question detail](#_8nq1lzmwentf) 26

[Change question content](#_ucamr5sormmp) 27

[Create a new question](#_1z6252ovcqxj) 27

[Validate submitted question](#_8xixxkp1og0f) 29

[Search/Filter question](#_euadviaqj158) 31

[View user information](#_uy1oqxexfq4x) 31

[Change user status](#_3ykvxxr1xrgs) 32

[Search/Filter user](#_udjje7x5vrs6) 32

[View exchanging item](#_oqc0tdmm2uvl) 33

[Create an exchanging item](#_g1elzv3e2k7n) 33

[Change the exchanging item information](#_64sy9c63toa4) 34

[Search/Filter item (Staff)](#_7mhtk72ycgiy) 35

[View personal information](#_m1eopbcqxcbt) 36

[Change personal information](#_dotz61p6p3jf) 36

[View upcoming event](#_roq5pwfcr68f) 38

[View ongoing event detail](#_25babtw2f0vf) 39

[Join ongoing event (Waiting room)](#_8vq0cxv75td) 39

[Join ongoing event (game room)](#_2o2scf34a2vu) 40

[Choose question in the game room](#_rv7qnrppuxwq) 41

[Filling the keyword in the puzzle](#_hfgay1y5jxpg) 41

[Submit the puzzle](#_69d3e89e0h0i) 42

[Leave the game room](#_6dakcx4jkqb) 42

[See the solution of a joined ongoing event](#_mn5wkrp0ovvn) 43

[See the ranking of a joined event](#_n7s13v3c4a6z) 43

[View joined event detail](#_rsg7gev6yjby) 44

[View Netcoin shop](#_124vhsfdbj74) 44

[View exchanging item detail](#_2s54j3nutdvi) 45

[Exchanging item](#_l7ztiofi3yi9) 45

[View transaction history](#_q9a38tqyiu4p) 46

[Search/Filter item (Player)](#_rkpezsbvybgq) 47

[Contribute question](#_9huicw6tb1e5) 47

[View contributed question](#_msjm86hbhnrr) 49

[**Logical component model**](#_2fk6b3p) **49**

[Authorization and Authentication](#_3ep43zb) 52

[Administration System](#_1tuee74) 52

[Question Bank Management System](#_4du1wux) 52

[Event Management System](#_2szc72q) 52

[Safety component](#_184mhaj) 52

[Business logic](#_3s49zyc) 53

[Adapter](#_279ka65) 53

[Connector](#_meukdy) 53

[Monitoring](#_36ei31r) 53

[Error handling](#_1ljsd9k) 53

[Validation](#_45jfvxd) 53

[Caching](#_2koq656) 53

[Setting component](#_zu0gcz) 53

[Text component](#_3jtnz0s) 53

[Date manager](#_1yyy98l) 54

[Logging](#_4iylrwe) 54

[Databases](#_2y3w247) 54

[Other systems](#_1d96cc0) 54

[**Business processes**](#_2ce457m) **55**

[Concepts](#_4anzqyu) 55

[Workflow](#_14ykbeg) 55

[Automatic steps](#_3oy7u29) 55

[Wait step](#_243i4a2) 56

[Processes](#_j8sehv) 56

[Changes from gameplay data](#_338fx5o) 56

[Changes from event data](#_1idq7dh) 56

[**Data**](#_11si5id) **56**

[File management](#_3ls5o66) 57

[Information model](#_20xfydz) 57

[Data model for batch and operations administration](#_4kx3h1s) 57

[User](#_1f7o1he) 59

[Player](#_yv640ikalku) 59

[Participant](#_zb96udtplbs2) 59

[Event](#_lw6sx7dhlcqz) 60

[Question](#_lw6sx7dhlcqz) 60

[Item](#_lw6sx7dhlcqz) 61

[Transaction](#_lw6sx7dhlcqz) 61

[Standard software](#_2jh5peh) 63

[IntelliJ IDEA](#_9mdsbs54ouwh) 63

[Java 15.0.1](#_ymfzma) 63

[NPM (Node Package Manager)](#_b88u08bmyeic) 63

[Springboot](#_1xrdshw) 63

[Hibernate](#_4hr1b5p) 63

[ReactJS](#_s9xrqeuxgrmd) 64

[NodeJS](#_im522gpdq1q0) 64

[PostgreSQL](#_c9uhgm58hqht) 64

[Rules and system parameters](#_3fg1ce0) 64

[System parameters](#_1ulbmlt) 64

[Evaluation](#_4ekz59m) 65

[History](#_18vjpp8) 65

[Rolling out](#_3sv78d1) 66

[Rules](#_280hiku) 67

[Evaluation](#_n5rssn) 67

[Rules sheet](#_1maplo9) 68

[History](#_2lfnejv) 69

[Versioning](#_3kkl7fh) 70

[Rolling out](#_1zpvhna) 70

[**Deployment**](#_1e03kqp) **70**

[**Performance**](#_2d51dmb) **72**

[Capacity](#_sabnu4) 72

[Usability](#_3c9z6hx) 72

[Security](#_1rf9gpq) 72

[**Security architecture**](#_24ufcor) **73**

[Authentication](#_jzpmwk) 73

[Authorization](#_33zd5kd) 73

[Security Measures](#_1j4nfs6) 73

[Application security](#_2i9l8ns) 74

# Introduction

This document is part of the system documentation of the Netcompany Crossword Puzzle (NCP).

## Purpose

The purpose of this document is to describe the software architecture. Ie. which components are included in the solution, and how are the components that have been developed as part of the Netcompany Crossword Puzzle (NCP) designed and developed. The document also describes essential architectural principles for eg logging, caching, security management, etc.

## Audience

The document is primarily aimed at architects, developers and operating personnel who need insight into how the solution is implemented.

## Definitions and abbreviations

|  |  |
| --- | --- |
| Designation | Description |
|  |  |
|  |  |
|  |  |

## Reading instructions

The document can be read alone, but can possibly be supplemented with [O0400] (O0400 can advantageously be read first). The document can also be used as a reference work for people who want specific knowledge about eg principles of logging or security.

## Restrictions and outstanding

The table below contains information on limitations and outstanding issues that require clarification.

|  |  |
| --- | --- |
| Subject | Description |
| Detailed processes | Details of the individual processes, what initiates them, etc. are detailed in the development process that contains the process in question. Thus, only the overall transverse characteristics of processes are described herein. |
| Logging of unauthorized access | Implementation of the details of attempts to identify unauthorized access attempts will only be detailed in later development processes, as it is still unclear how aggressive this should be. |
| Detailing of implementation | Section 8 regarding implementation is for some components described very broadly. Work will continue on that during those deliveries. |

# Solution overview

This section describes the solution on a very general level. The content of the section will be detailed in later sections.

## Technical solution overview

This section provides an overview of components included in the solution, as well as their interdependencies (at the logical level). Details about ATP systems as well as the external systems can be found in [D0180] and further information about the technical infrastructure in [O0400].

**IntelliJ IDEA**

Version: 2020.2.3

Website: https://www.jetbrains.com/idea/

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

**Java 15.0.1**

Version: 15.0.1

Website: https://www.oracle.com/java/technologies/javase/15-0-1-relnotes.html

Java is the basic programming language for the system. Introduced in 1995, Java is widely used in many Enterprise solutions.

**NPM (Node Package Manager)**

Version: 6X

Website: https://docs.npmjs.com/cli/v6/commands/npm-install

NPM (Node Package Manager) is a package manager for the JavaScript programming language. npm, Inc. is a subsidiary of GitHub, an American multinational corporation that provides hosting for software development and version control with the usage of Git. It is the default package manager for the JavaScript runtime environment Node.js.

**Springboot**

Version: 2.4.1

Website: https://spring.io/projects/spring-boot

Spring is a frequently used framework that provides a large, coherent tool portfolio for Java applications. Spring contributes with, among other things, IoC and lays the foundation for a number of other areas, e.g. compared to JPA, batch and AOP.

**Hibernate**

Version: 5.4

Website: https://hibernate.org/orm/releases/

Hibernate is an object-relational mapping (ORM) framework for Java. It handles the persistence of Java class content into relational databases. The JPA specification is used with Hibernate as provider.

**ReactJS**

Version: 17.0.1

Website: https://reactjs.org/blog/2020/10/20/react-v17.html

ReactJS is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies

**PostgreSQL**

Version: 13.1

Website: https://www.postgresql.org/download/

PostgreSQL is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley.

## Functional solution overview

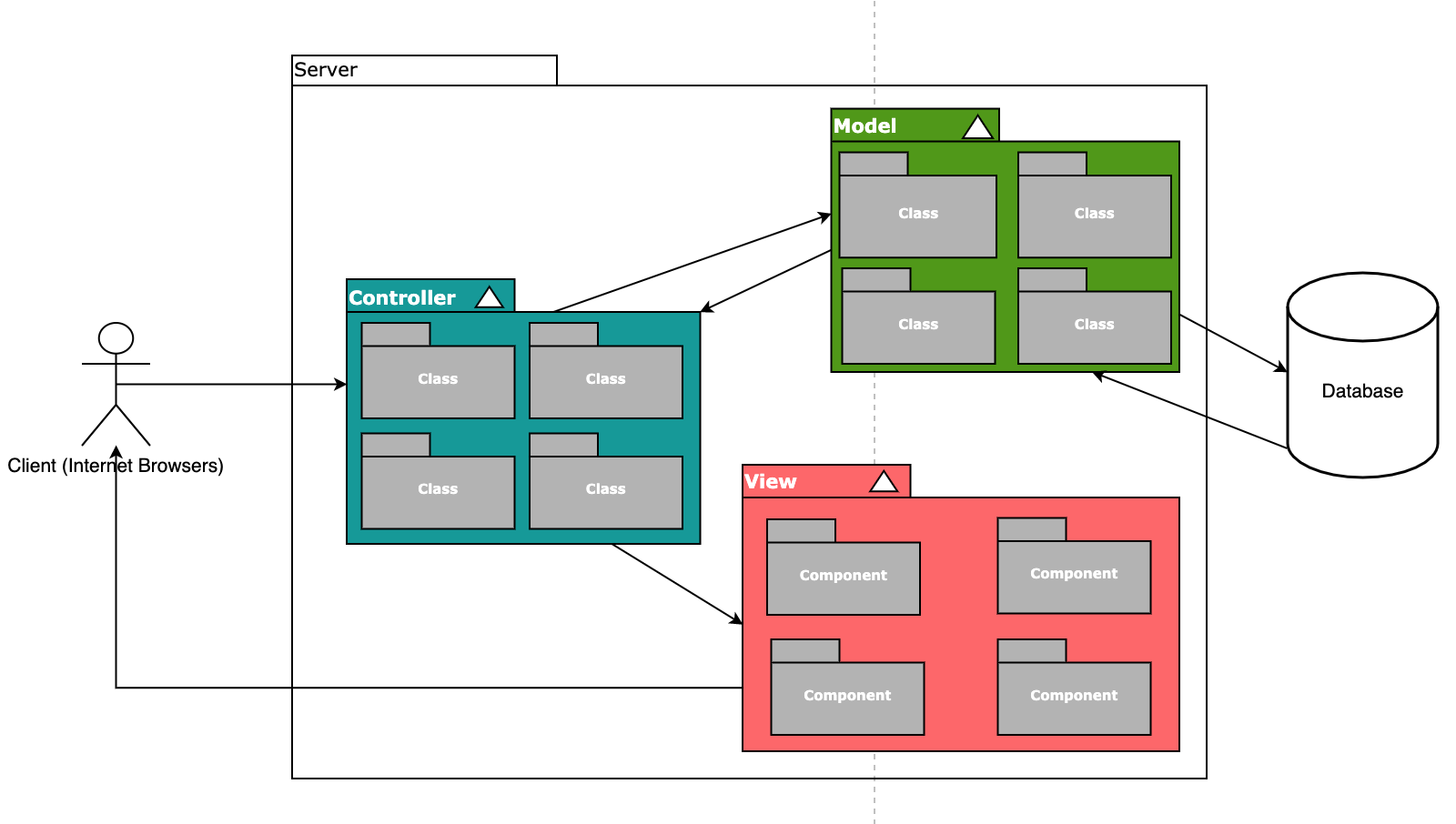


Figure 1: System Overview

Figure 1 illustrates the systems involved. The individual components will be detailed further in later sections.

The logical execution environment describes the relationship between the components in the solution without taking into account the physical device on which the components are executed. This abstraction makes it possible to avoid repeating the description for each environment.

* Component diagram – a UML diagram of the components in the solution, ideally with named dependencies/interfaces
* Component descriptions – description of the individual components in the component diagram
* Component dependencies – description of the individual dependencies/interfaces in the component diagram

If the architecture is very complex, a top-down breakdown with one description per subsystem is recommended.

**Model**

The model component stores data and its related logic. It represents data that is being transferred between controller components or any other related business logic (application business logic, validation logic, and database access logic). Models allow basic CRUD operations (create, read, update, delete).

It responds to the request from the views and also responds to instructions from the controller to update itself.

The model consists of many classes (user, question, event, etc), each model is the representative of an entity or a collection within the database

**View**

The view component represents the actual websites for the solution and consists of many smaller components.

Views request the models to give information to the controllers and from there to the views so that it resents the output presentation to the user and are created by the data collected from the model data. Some views will be reused on many pages / screens of the application (form, navbar, etc). Interacting with the views will trigger their corresponding controller.

A web page / screen is generated by many components of the views.

**Controller**

The controller component is responsible for controlling the way that a user interacts with an MVC application, it determines what response to send back to the views when a user makes a browser request.

It also sends commands to the model to update its state and sends commands to its associated view to change the view's presentation.

The controller consists of many classes. Each model will have its own corresponding controller.

# Architectural principles

This section describes the principles used in the development of the software architecture. Later sections describe how essential principles are realized in the design of the Netcompany Crossword Puzzle (NCP) platform.

## Modules

**The solution is divided into components (modules - Models, Views, Controllers), which communicate with each other through well-defined interfaces.**

The solution is thus divided into smaller and manageable parts (components), which ensures a better overview and project structure, better folders and files division, better division of responsibilities and simpler maintenance. This makes it possible to maintain and continuously update the individual components. See further information in **O0400 - Technical Infrastructure.**

## Loose clutch

**The components (modules) are loosely coupled to ensure that the components do not have strong transverse dependencies. See the A0100 section 8.3 - Project Dependencies for more information.**

The individual components of the solution are disconnected from each other through the division of responsibilities for the individual layers. For example, a business component does not depend on the characteristics of a given integration component, only on the data returned. Thus, there is no direct dependency which makes it easier to replace one integration component with another that complies with the same interface.

## Cohesion

**The components (modules) of the solution are coherent, which ensures that one functionality is not spread over multiple components.**

The individual sub-components of components all have an isolated area of ​​responsibility and together form one unit that handles one given functional area.

## Software

This section deals with the principles that apply to the core of the NCP services (for both users and HR staff).

### Data validation

**All input will be validated before it is processed.**

Before the service core starts processing on the basis of given input data, a data validation is performed in the business core. This is done to ensure that only data that is valid in the situation in question is operated on. The validation helps to reduce the risk of logical errors.

### Data security

**The system only allows a user to access the data relevant and approved for the user.**

It is the responsibility of the system to ensure that users see only the data to which they are entitled.

* Users (players) can only see their own registered information and relevant gameplay information.
* HR staff / Head administrators can gain access to all relevant information of the NCP and all actions will be logged in the database.

### Network security

**Network security must guarantee and ensure that unauthorized persons do not have the opportunity to call against the system or interact with .**

Firewalls and related security measures ensure that it is not possible to call directly into the services.

### Performance

**The solution must be designed for high measurable performance.**

The system must deliver high performance, have a simple, scalable design and at the same time be able to handle high peak loads in an appropriate way, where other systems are protected and optimized against overload..

### Availability

**The system must ensure high availability.**

It must be possible to distribute the load on multiple servers so that there is a high load tolerance that can ensure availability. Batch and subject systems must not be able to load each other, so that batch runs prevent work on the subject system.

### Robustness

**The system must be robust to inaccessible systems. The system must be fault tolerant.**

The system must be able to workaround when support and sub systems are not necessarily available. If an error occurs or a support system is unavailable, the system must be continued.

It must be ensured that there is no need for a manual process to re-establish connection to other underlying systems after underlying systems have been down and come up again (automatic handling of temporary downtime).

### Error handling

**The system must log all errors and roll back transactions.**

All errors must be registered. In the case of a transaction (activity) that is controlled by the system, it must be rolled back to the state before the transaction was started. An error message must be displayed to the user in a form of notification and for batch jobs it must be clear which entity or component is affected. By logging the data that is relevant in connection with tracking and identification of errors, fast error correction is ensured.

## Data

This section describes the principles for handling data in the system.

### Persistence

Data is maintained in the Netcompany local database. The majority of core data for the benefit core is retrieved via the MVC, but the HR staff and head administrators owns the following data:

#### History and logging

History of activities and interactions (user interactions with data).

#### Rules, texts and system parameters

Rules, texts and system parameters can be managed by the business administrator and persisted by the system.

### Database access

To ensure that all data operations in relational databases are performed in a secure manner, system users with limited rights are used. In addition, JDBC data sources are used, which are configured in the Java EE container. This helps ensure that data transferred between application servers and database servers does not leave the internal operating networks.

The database should not be written outside of system functions, as the caching and writing are managed in the application server.

## Test

This section describes the principles for testing the system.

### Unit test

All components should have automated device testing. The automatic unit tests hit common scenarios, as in practice it makes the best sense in terms of maintenance and error correction times.

### Component test

Testing of components is isolated so that there are no dependencies on surrounding systems. This ensures that tests can be run independently of uptime and data quality on other systems and thus that errors in other components do not affect a given component's test cases.

### Integration test

All components should be tested for integration. This ensures that the interaction between the components works and that there is a gap through to various systems.

## Drift

This section describes the principles of operating the system.

### Logging

Logging is an essential part of ensuring traceability in connection with readings and not least updating information. Furthermore, logging helps to facilitate troubleshooting and possibly error correction. The level of logging can be configured for each environment. In addition to call logs, the overview is facilitated through a change log, where the individual changes are categorized.

Overall, there are the following log types:

* Change log - persisted in the database (information about who create or change which data)
* Security log - containing information about who has (tried) accessing which data

### Surveillance

Performance for service calls, batch jobs, etc. monitored to ensure transparency and the ability to respond to health changes to the system before it becomes critical. For this purpose, e.g. at response times, etc. for the different systems.

## Modular user interface

This section describes how the concept of modular user interface implementation is implemented.

### Templates

Modules are built in the code using templates. A template is a template that can be instantiated and pasted on a given page and location. It is filled in with data that can be context-dependent, -independent or static, which must be defined for the individual template. A template can also be rights managed quite specifically, so that e.g. only certain roles are given access and / or there must be rights to the data displayed. Templates can be reused in other templates or inserted individually in a screenshot.

# Use cases

This section provides an overview of the system from a user point of view by describing actors and significant use cases.

The use case model is specified in detail with side sketches in [D0160], it gives a complete picture of the system from the user's point of view.

Functionality performed by the system is described in section 6 Business processes, as well as [O0200 - Batch Jobs].

The figure below illustrates the actor and significant use cases for the NCP system, including 3 actors: User, Player, Staff/Admin.

Player and Staff/Admin are the extended actors of the User actor.

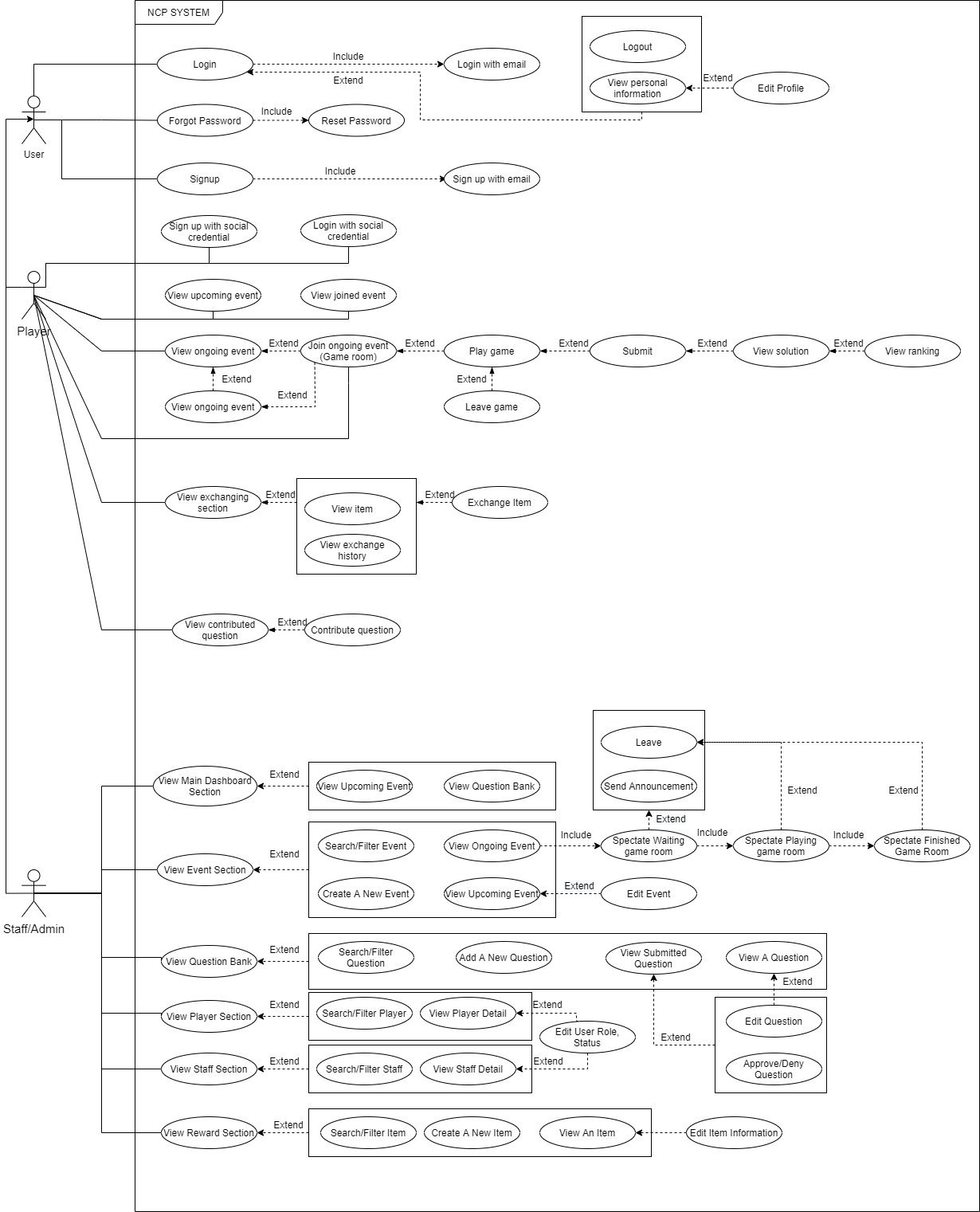


Figure 3 - NCP System Use Case Diagram

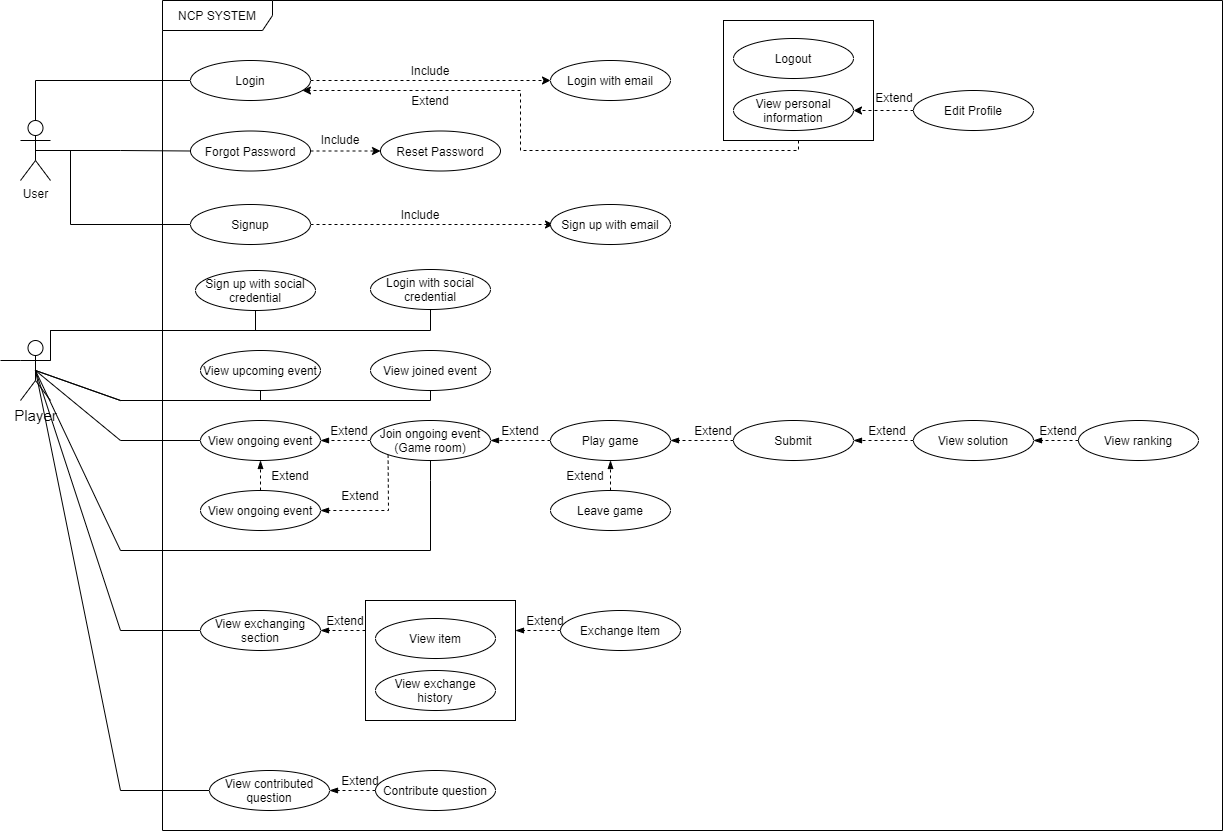


Figure 3.1 - NCP System Use Case Diagram (Player)



Figure 3.2 - NCP System Use Case Diagram (Admin/Staff)

### 

## Login

### Login with email

|  |  |
| --- | --- |
| **Use case 1: Login with email** | |
| Purpose: | To access the NCP system |
| Actors: | Player, Staff |
| Description: | Actor logins to the NCP system with his/her account by using email. |
| Starting conditions: | The actor must have an email account that are signed up (verified by the system) |
| Main flow: | 1. The actor fills in the data fills    * The email input (abcxyz@mail.com)    * The password input (\*\*\*\*\*\*) 2. Click in the “Sign in” button 3. Access into the system |
| Alternative flows: | 1. Click in the “Sign in button”    1. The system alerts that the inputs fields are not correct (wrong email or wrong password)    2. Redo step 1 and 2 (Main flow) until the email and the password are correct. 2. Access into the system |
| Final conditions: | The actor accesses the NCP system with their verified account and with their role. |

### Login with social credentials

|  |  |
| --- | --- |
| **Use case 2: Login with social credentials** | |
| Purpose: | To access the NCP system |
| Actors: | Player |
| Description: | Actor logins to the NCP system with his/her account by using social credentials. |
| Starting conditions: | The actor must have an social credentials account that are signed up (verified by the system) |
| Main flow: | 1. The actor clicks on the “continue with Facebook” or the “continue with google” button. 2. The page leads to the Facebook/Google sign in page. 3. The actor fills in the inputs.    * The email input (abcxyz@mail.com)    * The password input (\*\*\*\*\*\*) 4. The actor clicks on the sign in button. 5. The page redirects back to the NCP system and access to the system. |
| Alternative flows: | Alternative flow 1:   1. Click in the “Sign in button”    * The system alerts that the inputs fields are not correct (wrong email or wrong password)    * Redo step 3 and 4 (Main flow) until the email and the password are correct. 2. The page redirects back to the NCP system and access to the system.   Alternative flow 2:   1. The page leads to the Facebook/Google approval sign page. 2. The actor clicks on the approve button. 3. The page redirects back to the NCP system and access to the system. |
| Final conditions: | The actor accesses the NCP system with their verified account and with his/her role. |

### Logout

|  |  |
| --- | --- |
| **Use case 3: Logout** | |
| Purpose: | To leave the NCP system |
| Actors: | Player, Staff |
| Description: | The actor logouts of the NCP system. |
| Starting conditions: | The actor must have signed in the system |
| Main flow: | 1. The actor clicks on the personal profile section 2. Click on the “Logout” button 3. The actor clicks “confirm” button 4. The system redirect back to the Login page |
| Alternative flows: | 1. Click on the “cancel” button    1. Back to the personal profile section 2. The actor does not logout the system |
| Final conditions: | The actor logouts the system successfully |

## Sign up

### Sign up with Email

|  |  |
| --- | --- |
| **Use case 4: Sign up with email** | |
| Purpose: | To have an account in order to access the NCP system |
| Actors: | Player, Staff |
| Description: | The actor create an account by using email |
| Starting conditions: | The actor must have an email account (gmail, yahoo, ...) |
| Main flow: | 1. The actor clicks on the “Sign Up” button 2. The sign up section popped up 3. The actor fills in the input fields    * The email input (abcxyz@mail.com)    * The password input (\*\*\*\*\*\*)    * The password confirm input (\*\*\*\*\*\*) 4. The actor clicks on the “Sign Up” button 5. The actor accesses into the NCP system |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Sign Up” button    1. The fields are not correct (email are not valid or password confirm is not the same as the password field)    2. Redo step 3 and 4 (Main flow) until the fields are valid. 2. The actor accesses into the NCP system   Alternative flow 2:   1. Click on the “cancel” button    1. Back to login page 2. The actor does not sign up |
| Final conditions: | The actor successfully creates an account by using email |

### Sign up with social credentials

|  |  |
| --- | --- |
| **Use case 5: Sign up with social credentials** | |
| Purpose: | To access the NCP system |
| Actors: | Player |
| Description: | Actor logins to the NCP system with his/her account by using social credentials. |
| Starting conditions: | The actor must have an social credentials account that are signed up (verified by the system) |
| Main flow: | 1. The actor clicks on the “continue with Facebook” or the “continue with google” button. 2. The page leads to the Facebook/Google sign in page. 3. The actor fills in the inputs.    * The email input (abcxyz@mail.com)    * The password input (\*\*\*\*\*\*) 4. The actor clicks on the sign in button. 5. The page redirects back to the NCP system and access to the system. |
| Alternative flows: | Alternative flow 1:   1. Click in the “Sign in button”    1. The system alerts that the inputs fields are not correct (wrong email or wrong password)    2. Redo step 3 and 4 (Main flow) until the email and the password are correct. 2. The page redirects back to the NCP system and access to the system.   Alternative flow 2:   1. The page leads to the Facebook/Google approval sign page. 2. The actor clicks on the approve button. 3. The page redirects back to the NCP system and access to the system. |
| Final conditions: | The actor successfully creates an account via social credentials |

## Find password if forgot

|  |  |
| --- | --- |
| **Use case 6: Find password if forgot** | |
| Purpose: | To find the password if forgot |
| Actors: | Player, Staff |
| Description: | The actor finds the password of an existing account |
| Starting conditions: | The actor must have an verified account of the NCP system |
| Main flow: | 1. The actor clicks on the ”Forgot Password” link in the login page.    1. The forgot password section popped up 2. The actor fills in the email input.    * The email input (abczyx@mail.com) 3. The actor clicks on the “Send” button    1. The verification code sends to the corresponding email 4. The page actor fills the code in the code input 5. The actor clicks on the “Submit button” 6. The page leads to the change password page 7. The actor fills in the fields:  * The password input (\*\*\*\*\*\*) * The confirm password input (\*\*\*\*\*\*)  1. The actor clicks on “Change Password” button 2. The page redirects back to the login page |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Submit button”    1. The system alerts that the input code is not correct    2. Redo step 3 and 4 (Main flow) until the confirmed code is correct..   Alternative flow 2:   1. The actor clicks on “Change Password” button    1. The system alerts that the confirm password and the password are not the same    2. Redo step 7 and 8 (Main flow) until password and the password are the same 2. The page redirects back to the login page |
| Final conditions: | The actor successfully reset the password |

## Overview current event

|  |  |
| --- | --- |
| **Use case 7: Overview current event** | |
| Purpose: | See the detail of the ongoing event |
| Actors: | Staff |
| Description: | The actor see all of the details of the ongoing event and can spectate the ongoing event |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on “Event” section on the staff dashboard menu    1. The event section page appears 2. The actor clicks on any event in the ongoing event part    1. The event detail appears    2. The event detail page appears (waiting room) when the event starts.    3. After the waiting time is up, the game room appears with its details.    4. After the gaming time is up, the finished game room appears with its details |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on any event in the ongoing event part    1. The event detail page appears (waiting room) when the event starts.    2. After the waiting time is up, the game room appears with its details.    3. After the gaming time is up, the finished game room appears with its details   Alternative flow 2:   1. The actor clicks on any event in the ongoing event part    1. After the waiting time is up, the game room appears with its details.    2. After the gaming time is up, the finished game room appears with its details |
| Final conditions: | The actor can see all of the details of the ongoing event |

## Send announcement in waiting room

|  |  |
| --- | --- |
| **Use case 8: Send announcement in waiting room** | |
| Purpose: | Send announcement for the players in the waiting room of an ongoing event |
| Actors: | Staff |
| Description: | After joining an ongoing event and its waiting room, the actor can sends announcements for the player in that event |
| Starting conditions: | The actor must have an verified account of the NCP system,logged in the system and joins the waiting room of an ongoing event |
| Main flow: | 1. The actor clicks on the announcement bar and types the announcement on it 2. The actor clicks on the  button    1. The confirmation section popped up 3. The actors clicks on the “Yes” button    1. The announcement is sent |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the  button    1. The confirmation section popped up 2. The actors clicks on the “No” button    1. The announcement is not sent, the confirmation section disappears (the text in the announcement bars is still there) |
| Final conditions: | The actor can send the announcement to the players |

## Overview upcoming event

|  |  |
| --- | --- |
| **Use case 9: Overview upcoming event** | |
| Purpose: | See the detail of the upcoming event |
| Actors: | Staff |
| Description: | The actor finds the password of an existing account |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on “Event” section on the staff dashboard menu    1. The event section page appears with the upcoming event list 2. Click on any upcoming event in the section    1. The event details page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see all of the details of the ongoing event |

## Create event

|  |  |
| --- | --- |
| **Use case 10: Create event** | |
| Purpose: | To create an event for the player in the NCP system |
| Actors: | Staff |
| Description: | The actor create an event by choosing a core keyword and other related information |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Event” section on the staff dashboard menu    1. The event section page appears 2. The actor clicks on the “Create a new event” button    1. The create new event page appears 3. The actor fills in all the fields:  * Event name (Text) * Core keyword (max 10 characters, no special character and digit allow) * Category * Labels * Event description * Can select horizontal keyword * Date started * Date ended * Status (Draft, Publish)  1. The actor clicks on “Create” button    1. The confirmation section popped up 2. The actor clicks on the “Confirm” button    1. A new event created    2. Back to the Event overview page |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Create” button    1. The system alerts that the input fields are not valid (keyword overlap, fields are empty, ...)    2. Redo step 3 and 4 (Main flow) until the fields are valid.    3. The confirmation section popped up 2. The actor clicks on the “Confirm” button    1. A new event created    2. Back to the Event overview page   Alternative flow 2:   1. The actor clicks on “Cancel” button    1. The confirmation section popped up 2. The actor clicks on the “Confirm” button    1. The information are deleted    2. Back to the Event overview page   Alternative flow 2.1:   1. The actor clicks on “Cancel” button    1. The confirmation section popped up 2. The actor clicks on the “Cancel” button    1. Back to the create a new event page (with the information that have been filled in step 3 (Main flow))   Alternative flow 3:   1. The actor clicks on “Create” button    1. The confirmation section popped up 2. The actor clicks on the “Cancel” button    1. The information are deleted    2. Back to the Event overview page |
| Final conditions: | The actor successfully creates a new event |

## Search/Filter event

|  |  |
| --- | --- |
| **Use case 11: Search/Filter event** | |
| Purpose: | Search/Filter for events |
| Actors: | Staff |
| Description: | The actor finds the events by searching or filtering |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on “Event” section on the staff dashboard menu    1. The event section page appears 2. Filling in the search bar or choosing options in the filter bar 3. Press enter (If using search bar)    1. The matching events appear |
| Alternative flows: |  |
| Final conditions: | The actor can find the matching events |

## Overview question bank

|  |  |
| --- | --- |
| **Use case 12: Overview question bank** | |
| Purpose: | See questions in question bank |
| Actors: | Staff |
| Description: | The actor can see all the questions in the question bank |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Question bank” section on the staff dashboard menu    1. The question bank page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see question of the question bank |

## View question detail

|  |  |
| --- | --- |
| **Use case 13: View question detail** | |
| Purpose: | See the detail of the question in question bank |
| Actors: | Staff |
| Description: | The actor can see all the detail of the question in the question bank |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Question bank” section on the staff dashboard menu    1. The question bank page appears 2. Click on any question in the question list    1. The question detail page appears. |
| Alternative flows: |  |
| Final conditions: | The actor can see the question detail |

## Change question content

|  |  |
| --- | --- |
| **Use case 14: Change question content** | |
| Purpose: | Change the detail of the question in question bank |
| Actors: | Staff |
| Description: | The actor can see all the detail of the question in the question bank |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Question bank” section on the staff dashboard menu    1. The question bank page appears 2. Click on any question in the question list    1. The question detail page appears. 3. The actor changes the question description (max 40 words) 4. The actor clicks on the “Edit” button.    1. A confirm section popped up 5. The actor clicks on the “Save” button 6. The changed information are saved |
| Alternative flows: | 1. The actor clicks on the “Cancel” button 2. The changed information are canceled |
| Final conditions: | The actor can change the question detail |

## Create a new question

|  |  |
| --- | --- |
| **Use case 15: Create a new question** | |
| Purpose: | Create a new question |
| Actors: | Staff |
| Description: | The actor can create a new question |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Question bank” section on the staff dashboard menu    1. The question bank page appears 2. The actor clicks on the create new question    1. The add new question page appears. 3. The actor fills in the input and information  * Question description (text, max 40 words) * Answer (only text, no digit or special character, max 10 characters) * Category (Dropdown input) * Label (Typeahead type) * Difficulty (dropdown input , 1 - 10 scale)  1. The actor clicks on the “Create” button.    1. A confirm section popped up 2. The actor clicks on the “Save” button 3. A new question is added to the question bank |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Create” button.    1. The system alerts that the field inputs are not valid    2. Redo step 3, 4 (Main flow) until the conditions are satisfied 2. The actor clicks on the “Save” button 3. A new question is added to the question bank   Alternative flow 2:   1. The actor clicks on the “Create” button.    1. A confirm section popped up 2. The actor clicks on the “Cancel” button 3. The information are not saved |
| Final conditions: | The actor successfully creates a new question |

## Validate submitted question

|  |  |
| --- | --- |
| **Use case 16: Validate submitted question** | |
| Purpose: | To approve or deny a submitted question of a player |
| Actors: | Staff |
| Description: | The actor can see all of the information of the submitted question and then can approve or deny it |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on the “Question bank” section on the staff dashboard menu    1. The Question bank page appears 2. The actor click on any submitted question in the submitted question list    1. The question information page appears. 3. The actor selects the difficulty of the question by clicking on the difficulty field and then choose the difficulty (1 - 10) 4. The actor clicks on the “Approve” button.    1. The confirmation section popped up 5. The actor clicks on the “Confirm” button 6. A submitted question is added to the question bank |
| Alternative flows: | Alternative flow 1:   1. The actor selects the difficulty of the question by clicking on the difficulty field and then choose the difficulty (1 - 10) 2. The actor changes other fields:  * Category * Labels * Question description  1. The actor clicks on the “Approve” button.    1. The confirmation section popped up 2. The actor clicks on the “Confirm” button 3. A submitted question is added to the question bank   Alternative flow 1.1:   1. The actor selects the difficulty of the question by clicking on the difficulty field and then choose the difficulty (1 - 10) 2. The actor changes other fields:  * Category * Labels * Question description  1. The actor clicks on the “Approve” button.    1. The system alerts that the input fields are not valid    2. Redo step 4-5 until the conditions are satisfied    3. The confirmation section popped up 2. The actor clicks on the “Confirm” button 3. A submitted question is added to the question bank   Alternative flow 1.2:   1. The actor selects the difficulty of the question by clicking on the difficulty field and then choose the difficulty (1 - 10) 2. The actor changes other fields:  * Category * Labels * Question description  1. The actor clicks on the “Approve” button.    1. The confirmation section popped up 2. The actor clicks on the “Cancel” button    1. Back to the detail of the submitted question page   Alternative flow 2:   1. The actor selects the difficulty of the question by clicking on the difficulty field and then choose the difficulty (1 - 10) 2. The actor clicks on the “Approve” button.    1. The confirmation section popped up 3. The actor clicks on the “Cancel” button    1. Back to the detail of the submitted question page   Alternative flow 3:   1. The actor click on any submitted question in the submitted question list    1. The question information page appears. 2. The actor clicks on the “Approve” button.    1. The system alerts that the input fields are not valid    2. Redo step 4-5 until the conditions are satisfied    3. The confirmation section popped up 3. The actor clicks on the “Deny” button 4. A submitted question is denied |
| Final conditions: | The actor can see the user information |

## Search/Filter question

|  |  |
| --- | --- |
| **Use case 17: Search/Filter question** | |
| Purpose: | Search/Filter for questions |
| Actors: | Staff |
| Description: | The actor finds the questions by searching or filtering |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on “Question bank” section on the staff dashboard menu    1. The question bank section page appears 2. Filling in the search bar or choosing options in the filter bar 3. Press enter (If using search bar)    1. The matching questions appear |
| Alternative flows: |  |
| Final conditions: | The actor can find the matching questions |

## View user information

|  |  |
| --- | --- |
| **Use case 18: View user information** | |
| Purpose: | See users information of the system |
| Actors: | Staff |
| Description: | The actor can see all users of the NCP system |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Player” or “Staff” in the “User” section on the staff dashboard menu    1. The player/Staff page appears 2. The actor click on any user in the user list    1. The user information page appears. |
| Alternative flows: |  |
| Final conditions: | The actor can see the user information |

## Change user status

|  |  |
| --- | --- |
| **Use case 19: Change user status** | |
| Purpose: | To set the user status (Activate/Deactivate) |
| Actors: | Staff |
| Description: | The actor can set the user status |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Player” or “Staff” in the “User” section on the staff dashboard menu    1. The player/Staff page appears 2. The actor clicks on any user in the user list that he/she wanted to change.    1. The user information page appears. 3. The actor clicks on the status field.    1. The status field dropdowns 2 options:       * Activate       * Deactivate    2. The actor choose an option 4. The actor clicks on the “Edit” button.    1. The confirm section popped up 5. The actor clicks on the “Save” button. 6. Information saved. Back to the user information page |
| Alternative flows: | 1. The actor clicks on the “Edit” button.    1. The confirm section popped up 2. The actor clicks on the “Cancel” button. 3. Change canceled. Back to tube user information page |
| Final conditions: | The actor can set the user status |

## Search/Filter user

|  |  |
| --- | --- |
| **Use case 20: Search/Filter question** | |
| Purpose: | Search/Filter for player or staff |
| Actors: | Staff |
| Description: | The actor finds the users by searching or filtering |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on “Player“ or “Staff” section below the “User” section on the staff dashboard menu    1. The corresponding section page appears 2. Filling in the search bar or choosing options in the filter bar 3. Press enter (If using search bar)    1. The matching users appear |
| Alternative flows: |  |
| Final conditions: | The actor can find the matching users |

## View exchanging item

|  |  |
| --- | --- |
| **Use case 21: View exchanging item** | |
| Purpose: | See item information |
| Actors: | Staff |
| Description: | The actor can see the detail information of the item |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Reward” section on the staff dashboard menu    1. The Reward page appears 2. The actor clicks on any item in the item list    1. The item information page appears. |
| Alternative flows: |  |
| Final conditions: | The actor can see the item information |

## Create an exchanging item

|  |  |
| --- | --- |
| **Use case 22: Create an exchanging item** | |
| Purpose: | See users information of the system |
| Actors: | Staff |
| Description: | The actor can see all users of the NCP system |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Reward” section on the staff dashboard menu    1. The Reward page appears 2. The actor clicks on the “New Item” button    1. The create new item page appears. 3. The actor fills in the input fields:  * Item name (text) * Item price (number) * Item description (text) * Quantity (number) * Status (In stock, out of stock ) * Photos  1. The actor clicks on the “Add” button    1. A confirmation section popped up 2. The actor clicks “Confirm” button 3. A new item is added to the system |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Add” button    1. The system alerts that the input fields are not valid    2. Redo step 3-4 (Main flow) until the conditions are satisfied    3. A confirmation section popped up 2. The actor clicks “Confirm” button 3. A new item is added to the system   Alternative flow 1.1:   1. The actor clicks on the “Add” button    1. The system alerts that the input fields are not valid    2. Redo step 3-4 (Main flow) until the conditions are satisfied    3. A confirmation section popped up 2. The actor clicks “Cancel” button    1. Back to the create new item with the data in step 3   Alternative flow 2:   1. The actor clicks on the “Add” button    1. A confirmation section popped up 2. The actor clicks “Cancel” button    1. Back to the create new item with the data in step 3 |
| Final conditions: | The actor successfully create a new exchanging item |

## Change the exchanging item information

|  |  |
| --- | --- |
| **Use case 23: Change exchanging item information** | |
| Purpose: | Change the item information |
| Actors: | Staff |
| Description: | The actor can change the detail information of the item with the new information |
| Starting conditions: | The actor must have an verified staff account and login to the NCP system |
| Main flow: | 1. The actor clicks on “Reward” section on the staff dashboard menu    1. The Reward page appears 2. The actor clicks on any item in the item list    1. The item information page appears. 3. The actor changes the specific information 4. The actor clicks on the “Edit” button    1. The confirmation section popped up 5. The actor clicks the “Confirm” button 6. The new information of the item is saved |
| Alternative flows: | Alternative flow 1:   1. The actor changes the specific information 2. The actor clicks on the “Edit” button    1. The system alerts that the input fields are not valid    2. Redo step 3-4 (Main flow) until the conditions are satisfied    3. The confirmation section popped up 3. The actor clicks the “Confirm” button 4. The new information of the item is saved   Alternative flow 1.1:  The actor changes the specific information   1. The actor changes the specific information 2. The actor clicks on the “Edit” button    1. The system alerts that the input fields are not valid    2. Redo step 3-4 (Main flow) until the conditions are satisfied    3. The confirmation section popped up 3. The actor clicks the “Cancel” button 4. Back to the item information page with the changed information in step 3 but not saved   Alternative flow 2:   1. The actor changes the specific information 2. The actor clicks on the “Edit” button    1. The confirmation section popped up 3. The actor clicks the “Cancel” button 4. Back to the item information page with the changed information in step 3 but not saved |
| Final conditions: | The actor can change the item information |

## Search/Filter item (Staff)

|  |  |
| --- | --- |
| **Use case 24: Search/Filter item (Staff)** | |
| Purpose: | Search/Filter for items |
| Actors: | Staff |
| Description: | The actor finds the items by searching or filtering |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on “Reward” section on the staff dashboard menu    1. The “Reward” section page appears 2. Filling in the search bar or choosing options in the filter bar 3. Press enter (If using search bar)    1. The matching items appear |
| Alternative flows: |  |
| Final conditions: | The actor can find the matching items |

## View personal information

|  |  |
| --- | --- |
| **Use case 25: View personal information** | |
| Purpose: | To see the personal information |
| Actors: | Staff, Player |
| Description: | The actor can see his/her personal information |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile photo in the top right corner    1. The option panel appears at the right part of the page 2. The actor clicks on “More Info”    1. The detailed personal information page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the his/her personal information |

## Change personal information

|  |  |
| --- | --- |
| **Use case 26: Change personal information** | |
| Purpose: | To change the personal information |
| Actors: | Staff, Player |
| Description: | The actor can change his/her personal information |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile photo in the top right corner    1. The option panel appears at the right part of the page 2. The actor clicks on “More Info”    1. The detailed personal information page appears 3. The actor changes the input fields 4. The actor clicks on the “Save” button    1. The confirmation section popped up 5. The actor clicks on the “Confirm” button    1. The changed information are saved    2. Back to the personal information page |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Edit” button    1. The system alerts that the input fields are not valid    2. Redo step 3-4 (Main flow) until the conditions are satisfied    3. The confirmation section popped up 2. The actor clicks on the “Confirm” button    1. The changed information are saved    2. Back to the personal information page   Alternative flow 1.1:   1. The actor clicks on the “Save” button    1. The system alerts that the input fields are not valid    2. Redo step 3-4 (Main flow) until the conditions are satisfied    3. The confirmation section popped up 2. The actor clicks on the “Confirm” button    1. The changed information are saved    2. Back to the personal information page   Alternative flow 2:   1. The actor clicks on the “Save” button    1. The confirmation section popped up 2. The actor clicks on the “Cancel” button    1. Back to the personal information page with the changed information in step 3 (but not saved)   Alternative flow 3:   1. The actor clicks on the “Cancel” button    1. The confirmation section popped up 2. The actor clicks on the “Yes” button    1. Back to the personal information page with the old information (before step 3)   Alternative flow 3.1:   1. The actor clicks on the “Cancel” button    1. The confirmation section popped up 2. The actor clicks on the “No” button    1. Back to the personal information page with the changed information in step 3 (but not saved) |
| Final conditions: | The actor successfully change his/her personal information |

## View upcoming event

|  |  |
| --- | --- |
| **Use case 27: View upcoming event** | |
| Purpose: | To see the detailed upcoming event information |
| Actors: | Player |
| Description: | The actor can see the upcoming event of the NCP system |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the “Details” button of any event in the upcoming event list in the player main page    1. The corresponding upcoming event detail page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the upcoming event detail |

## View ongoing event detail

|  |  |
| --- | --- |
| **Use case 28: View ongoing event detail** | |
| Purpose: | To see the detailed ongoing event information |
| Actors: | Player |
| Description: | The actor can see the ongoing event of the NCP system |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor hovers(using desktop) or clicks (using smartphone/tablet) on any event in the ongoing event list in the player main page    1. The “Details” button appears 2. The actor clicks on the “Details” button    1. The event detail page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the ongoing event detail |

## Join ongoing event (Waiting room)

|  |  |
| --- | --- |
| **Use case 29: Join ongoing event (waiting room)** | |
| Purpose: | To joining the waiting room of the ongoing event |
| Actors: | Player |
| Description: | The actor joins the waiting room of the ongoing event of the NCP system |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the “Join Now” button of any event in the ongoing event list in the player main page    1. The waiting room of the event appears |
| Alternative flows: |  |
| Final conditions: | The actor successfully joins the waiting room of the ongoing event |

## Join ongoing event (game room)

|  |  |
| --- | --- |
| **Use case 30: Join ongoing event (game room)** | |
| Purpose: | To joining the game room of the ongoing event |
| Actors: | Player |
| Description: | The actor joins the gameroom of the ongoing event of the NCP system |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the “Join Now” button of any event in the ongoing event list in the player main page    1. The waiting room of the event appears    2. After the waiting time is up, the timer turns into the “Join” button 2. The actor clicks on the join button    1. The game room appears. |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Join Now” button of any event in the ongoing event list in the player main page    1. The waiting room of the event appears 2. The actor clicks on the “Leave” button    1. The confirmation section popped up 3. The actor clicks on the “Yes” button    1. Back to the home page   Alternative flow 1.1:   1. The actor clicks on the “Join Now” button of any event in the ongoing event list in the player main page    1. The waiting room of the event appears 2. The actor clicks on the “Leave” button    1. The confirmation section popped up 3. The actor clicks on the “No” button    1. The confirmation section disappears, back to the waiting room |
| Final conditions: | The actor successfully joins the game room of the ongoing event |

## Choose question in the game room

|  |  |
| --- | --- |
| **Use case 31: Choose question in the game room** | |
| Purpose: | To read the question and answer it |
| Actors: | Player |
| Description: | The actor joins the gameroom of the ongoing event of the NCP system and interacts with the puzzle by clicking on horizontal line of it and the the question appears |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and joined in a game room on an ongoing event |
| Main flow: | 1. The actor clicks on any horizontal line of the puzzle    1. The puzzle line that is clicked on will have blue background    2. The question appears in the keyword description |
| Alternative flows: |  |
| Final conditions: | The actor can see the question of that horizontal keyword line |

## Filling the keyword in the puzzle

|  |  |
| --- | --- |
| **Use case 32: Choose question in the game room** | |
| Purpose: | To fill keyword in the puzzle |
| Actors: | Player |
| Description: | The actor joins the gameroom of the ongoing event of the NCP system and interacts with the puzzle by filling keywords on it |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and joined in a game room on an ongoing event |
| Main flow: | 1. The actor clicks on any horizontal line of the puzzle    1. The puzzle line that is clicked on will have blue background    2. The question appears in the keyword description 2. The actor fill in the puzzle by typing each character |
| Alternative flows: |  |
| Final conditions: | The actor can see the question of that horizontal keyword line |

## Submit the puzzle

|  |  |
| --- | --- |
| **Use case 33: Submit the puzzle** | |
| Purpose: | To fill keyword in the puzzle |
| Actors: | Player |
| Description: | After joining a game room of an ongoing event and filling in the puzzle(optional), the player submit his/her answer |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and joined in a game room on an ongoing event |
| Main flow: | 1. The actor clicks on the “Submit” button    1. The confirmation section appears 2. The actor fill clicks on the “Yes” button    1. The answer submitted    2. The confirmation section disappears    3. Back to the game room but the “Submit” button disappears |
| Alternative flows: | 1. The actor fill clicks on the “Yes” button    1. The confirmation section disappears    2. Back to the game room with the filled in keyword |
| Final conditions: | The actor successfully submits the puzzle |

## Leave the game room

|  |  |
| --- | --- |
| **Use case 34:Leave game room** | |
| Purpose: | To leave the game room but the answer is saved (but not submitted) |
| Actors: | Player |
| Description: | After joining a game room of an ongoing event , the actor can leave the room anytime he/she wants but the answer is saved (but not submitted) |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and joined in a game room on an ongoing event |
| Main flow: | 1. The actor clicks on the “Leave” button    1. The confirmation section appears 2. The actor fill clicks on the “Yes” button    1. The answer is saved (but not submitted)    2. Back to the home page |
| Alternative flows: | 1. The actor fill clicks on the “No” button    1. The confirmation section disappears    2. Back to the game room with the filled in keyword |
| Final conditions: | The actor successfully submits the puzzle |

## See the solution of a joined ongoing event

|  |  |
| --- | --- |
| **Use case 35:See solution of a joined ongoing event** | |
| Purpose: | To see the joined ongoing event solution |
| Actors: | Player |
| Description: | After a joined ongoing event time is over, the actor can see the solution of that event |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and joined in a game room on an ongoing event |
| Main flow: | 1. After the time is up, the alert section popped up 2. The actor clicks on the “Solution” button    1. The game room with the solution and “Exit” button appear |
| Alternative flows: |  |
| Final conditions: | The actor can see the solution of the puzzle |

## See the ranking of a joined event

|  |  |
| --- | --- |
| **Use case 36:See the ranking of a joined ongoing event** | |
| Purpose: | To see the joined ongoing event ranking |
| Actors: | Player |
| Description: | After a joined ongoing event time is over, the actor can see the ranking of that event |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and join a game room on an ongoing event |
| Main flow: | 1. The actor fill clicks on the “Exit” button in the solution page of the event    1. The confirmation section popped up 2. The actor clicks on the “Yes” button    1. The ranking page appears |
| Alternative flows: | 1. The actor fill clicks on the “No” button    1. The confirmation section disappears |
| Final conditions: | The actor can see the ranking of the event |

## View joined event detail

|  |  |
| --- | --- |
| **Use case 37:View joined event** | |
| Purpose: | To see the joined event detail |
| Actors: | Player |
| Description: | The actor can see their event history and result |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and has joined in an event |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Event History” option    1. The event history page appears 3. The actor clicks on any joined event in the joined event list    1. The corresponding event detail page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the joined event detail page |

## View Netcoin shop

|  |  |
| --- | --- |
| **Use case 38: View Netcoin shop** | |
| Purpose: | To see the Netcoin shop |
| Actors: | Player |
| Description: | The actor can see the Netcoin shop and the exchanging items |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Netcoin shop” option    1. The Netcoin shop page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the Netcoin shop |

## View exchanging item detail

|  |  |
| --- | --- |
| **Use case 39: View exchanging item detail** | |
| Purpose: | To see the exchanging item details |
| Actors: | Player |
| Description: | The actor can see the exchanging items detail in the Netcoin shop |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Netcoin shop” option    1. The Netcoin shop page appears 3. The actor clicks on the photo or the “Info” button of any exchanging item in the Netcoin shop    1. The corresponding exchanging item detail page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the exchanging item details that he/she would like to see |

## Exchanging item

|  |  |
| --- | --- |
| **Use case 40: Exchanging item** | |
| Purpose: | To exchanging the item that the actor would like to have |
| Actors: | Player |
| Description: | The actor can exchanges an item in the Netcoin shop with the Netcoin he/she has |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Netcoin shop” option    1. The Netcoin shop page appears 3. The actor clicks on the “Exchange” button of any exchanging item in the Netcoin shop    1. The exchanging section of the corresponding item appears 4. The actor select the quantity of the item that he/she wants to exchange 5. The actor clicks on the “Exchange” button    1. The confirmation section appears 6. The actor clicks on “Yes” button    1. An alert popped up “You have exchanged items...” |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the photo or the “Info” button of any exchanging item in the Netcoin shop    1. The corresponding exchanging item detail page appears 2. The actor clicks on the “Exchange” button    1. The exchanging section appears 3. The actor select the quantity of the item that he/she wants to exchange 4. The actor clicks on the “Exchange” button    1. The confirmation section appears 5. The actor clicks on “Yes” button    1. An alert popped up “You have exchanged items...”   Alternative flow 2:   1. The actor clicks on the “Exchange” button    1. The system alert popped up that the Netcoin the actor has is not enough to exchange the item and come up with some advice for the player how to earn Netcoin 2. The actor clicks on the “Back” button    1. The system alert disappears    2. Back to the exchanging section   Alternative flow 3:   1. The actor clicks on the “Exchange” button    1. The confirmation section appears 2. The actor clicks on “No” button    1. The confirmation section disappears    2. Back to the exchanging section |
| Final conditions: | The actor successfully exchanges the item that he/she wants |

## View transaction history

|  |  |
| --- | --- |
| **Use case 41: View transaction history** | |
| Purpose: | To see the actor’s transaction history |
| Actors: | Player |
| Description: | The actor can see the transaction history |
| Starting conditions: | The actor must have an verified account of the NCP system, logged in the system and has exchanged an item |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Netcoin shop” option    1. The Netcoin shop page appears 3. The actor clicks on the “Purchase history” button at the top right corner    1. The transaction history page appears 4. The actor clicks on any transaction history in the list    1. The corresponding transaction detail page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the transaction history |

## Search/Filter item (Player)

|  |  |
| --- | --- |
| **Use case 42: Search/Filter item (Player)** | |
| Purpose: | To search/filter for items |
| Actors: | Player |
| Description: | The actor finds the items by searching or filtering |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Netcoin shop” option    1. The Netcoin shop page appears 3. Filling in the search bar or choosing options in the filter bar 4. Press enter (If using search bar)    1. The matching items appear |
| Alternative flows: |  |
| Final conditions: | The actor can find the matching items |

## Contribute question

|  |  |
| --- | --- |
| **Use case 43: Contribute question** | |
| Purpose: | To contribute a new question to the question bank of the NCP system |
| Actors: | Player |
| Description: | The actor contributes new question to the question bank |
| Starting conditions: | The actor must have an verified account of the NCP system and logged in the system |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Your question” option    1. The question contribution list page appears 3. The actor clicks on the “Contribute” button    1. The contribution page appears 4. The actor fills in the input fields:  * Question description (Text) * Keyword (Only text, no digit or special characters allowed, max 10 characters) * Category (dropdown type) * Labels (typeahead type)  1. The actor clicks on the “Submit” button    1. The confirmation section popped up 2. The actor clicks on the ”Yes” button    1. The confirmation section disappears    2. The new question has been submitted and waiting for review    3. Back to the question contribution list page |
| Alternative flows: | Alternative flow 1:   1. The actor clicks on the “Submit” button    1. The system alerts that the input fields are not valid    2. Redo step 4-5 (Main flow) until the conditions are satisfied    3. The confirmation section popped up 2. The actor clicks on the ”Yes” button    1. The confirmation section disappears    2. The new question has been submitted and waiting for review    3. Back to the question contribution list page   Alternative flow 1.1:   1. The actor clicks on the “Submit” button    1. The system alerts that the input fields are not valid    2. Redo step 4-5 (Main flow) until the conditions are satisfied    3. The confirmation section popped up 2. The actor clicks on the ”No” button    1. The confirmation section disappears    2. Back to the contribution page with the same input information in step 4 (Main flow)   Alternative flow 2:   1. The actor clicks on the “Submit” button    1. The confirmation section popped up 2. The actor clicks on the ”No” button    1. The confirmation section disappears    2. Back to the contribution page with the same input information in step 4 (Main flow) |
| Final conditions: | The actor successfully contribute a new question to the question bank of the NCP system |

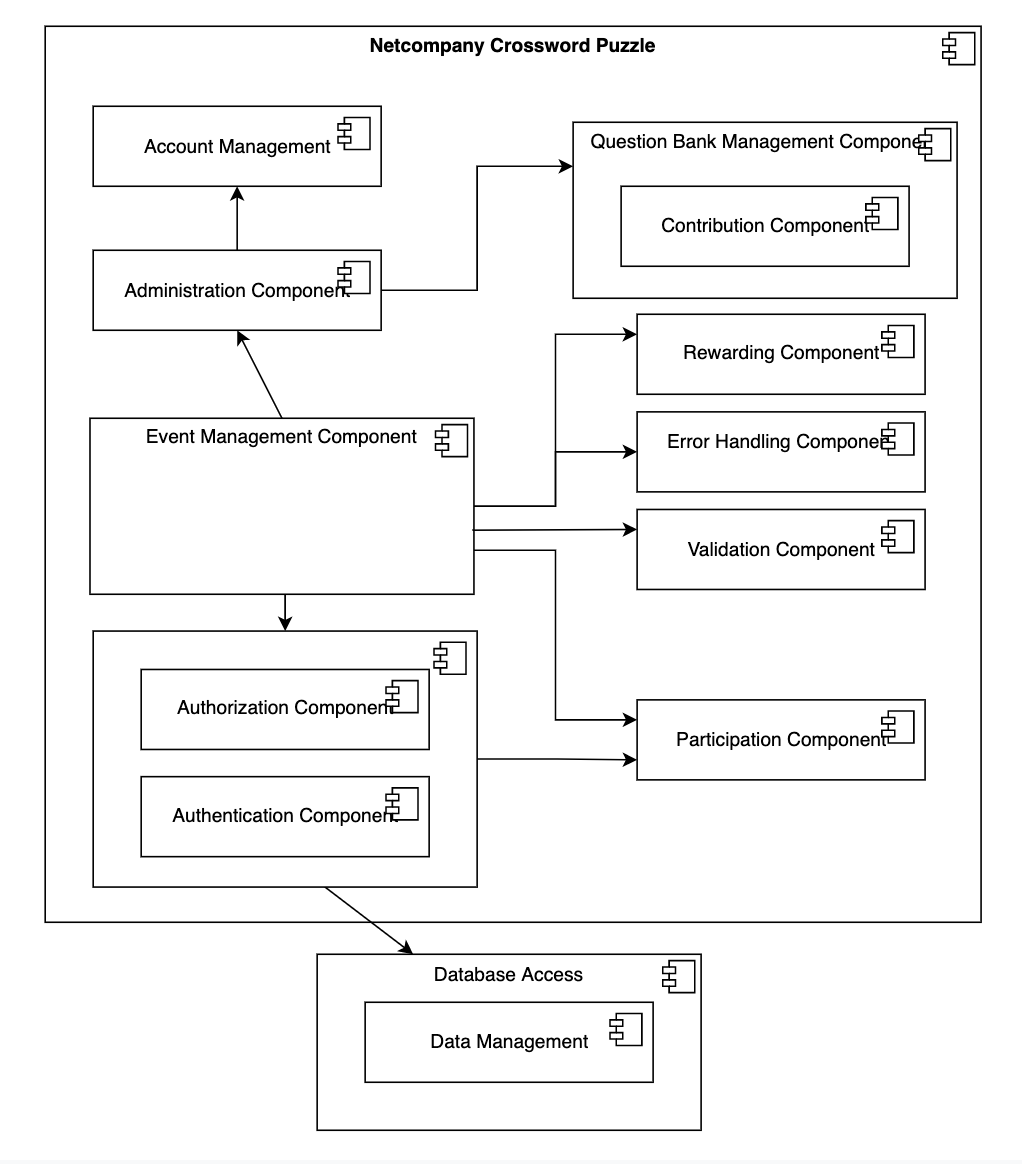
## View contributed question

|  |  |
| --- | --- |
| **Use case 44: View contributed question** | |
| Purpose: | To see a contributed question detail |
| Actors: | Player |
| Description: | The actor sees a contributed question detail |
| Starting conditions: | The actor must have an verified account of the NCP system,logged in the system and has contributed a question to the question bank |
| Main flow: | 1. The actor clicks on the profile picture at the top right corner    1. The option panel appears 2. The actor clicks on the “Your question” option    1. The question contribution list page appears 3. The actor clicks on the “Contribute” button    1. The contribution page appears 4. The actor clicks on any question in the contributed question list    1. The corresponding question detail page appears |
| Alternative flows: |  |
| Final conditions: | The actor can see the contributed question detail |

# Logical component model

This section describes the logical component model. Ie. the sub-components that are part of the subject system, self-service for citizens, batch settlement, etc.

The components are described at a general level, and it is explained what function they have, but it is not described how they are implemented. Significant details regarding. implementation is described later in section 7.



**Figure 4 -** Logical Component model

The figure above shows the main components of the performance core as well as the dependencies on other components. The individual components are described in the following subsections.

## Authorization and Authentication

The components also restrict access to data and the software used to manipulate that data.

The two components are used to authenticate and grant authorization to users who need to access digital information of the NCP.

Users are following a set of policies to decide who gets access to restricted data from any physical location. For example, HR staff only grants access to authorized users who have been registered in the system.

## Administration Component

The administration component, which is used only by the HR staff and admins, will have access to all other components of the NCP.

This also include the view access to all activities log (transactions by users) and traceback errors if needed

* + 1. **Account Management**

The role of the component is to monitor all registered users of the NCP system. Including their roles and the access right to the datasource.

## Question Bank Management Component

The management of the question bank can only be accessed by the HR staff and admins, including the following action

* Add a question
* Edit a question
* Approve / Decline a question submitted by users.
  + 1. **Contribution Component**

The component is responsible for handling questions contributed by the players. Each question will be validated by the staff. Comments and notifications will be further sent to the users afterward.

A player can contribute up to 3 questions per day and max 20 questions per month. Information about total contributed questions and their relevant data will be provided to the players.

Players with approved questions will then be rewarded a particular amount of Netcoin based on the difficulty of the questions.

## Event Management Component

The main service of the product is the event management component, which is also the core of the overall system.

The event management component is responsible for handling its related component in order to be effectively executed. This includes handling input from users (both HR staff and players) and output to both during the duration of an event.

* + 1. **Rewarding Component**

The rewarding system is responsible for calculating the netcoin earned by the players, which is determined and called by the following action

* Number of correct answers (including the core keyword) after an event is finished
* A question submitted by a player is approved by the HR staff. The reward will be based on the question’s difficulty

The event management system will use the rewarding system at the end of an event to distribute netcoins to the players.

* + 1. **Participation Component**

The participation system handles data of users joining an event, which will be written immediately into the database and sent to the event management system. The output data will then be shown on both staff and players screen.

### Error handling

The purpose of the error handling component is to ensure a uniform handling of errors, so that there is always the same interface in relation to error situations. The component defines known types of errors and their severity.

### Validation

The validation component is responsible for validation of incoming data. The input data is validated against the rules set up for its related entity, for example that the answer of a question must consist of only alphabet letters and max length is 10.

* 1. **Data Access**

The databases contain the information specific to the service core. The data that does not exist in other source systems but is necessary to support the processes that the system supports is found in the databases. The databases also contain logging information, settings and the persistent rules on which the rule engine is based.

* + 1. **Data Management**

The component is responsible for storing and retrieving users data, that includes acquiring, validating, storing, protecting, and processing required data to ensure the accessibility, reliability, and timeliness of the data for its users.

* 1. **Other Components**

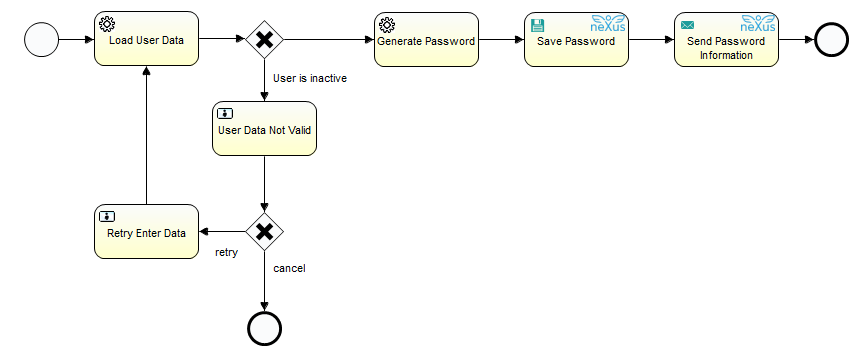
Other systems cover the other systems internally or externally, which are integrated with. These are further described in [D0180].

# Business processes

Business processes are modeled using *Workflows* , which are adapted to the needs of the solution. A workflow describes a flow, as well as controls the condition. A workflow can be started manually, started from another workflow, or as a result of a batch run. Workflows with time-dependent inputs are started from a batch run.

Workflows can be edited with a standard BPMN 2.0 tool, in this project we will use IntelliJ IDEA.

Figure 5 shows an example of what the workflow looks like during reset password. (Not an actual workflow, but a prototype)



**Figure 5** Example of process

## Concepts

This section describes the concepts used in the FY process engine, as well as their function.

### Workflow

A BPM workflow, modeled as a subset of BPMN 2.0. In relation to solution flows, a workflow can extend over several solution flows, but will only be a subset of the individual solution flow. A workflow has just one starting step. A workflow always ends in one or more final steps. A workflow links a series of steps, and can have conditional (from conditional steps) or parallel (from start workflow steps) branches. Multiple workflows can use the same step.

It is possible to add steps or change the order, but special attention must be paid to automatic steps. A step can expect special prerequisites to be met before it is called.

### Automatic steps

Steps cover actions in a workflow. In relation to solution flows, steps are concrete realizations of activities. Automatic steps are steps that are executed without delay.

1. **Reading steps** : steps that read to context, but which are idempotent.
2. **Write step** : step, which writes results to database or integrations.
3. **Error step** : step that handles error situation.
4. **Letter step** : merges letter and sender.
5. **Conditional step** : selects the way forward based on conditions .
6. **Start workflow step** : allows you to start another independent workflow.
7. **Start step** : Entrance to the workflow. Starting steps do nothing but start the workflow.
8. **End step** : End of workflow. There can be several possible end locations in a workflow. End Step updates the status of the workflow to be completed.

### Wait step

Steps cover actions in a workflow. In relation to solution flows, steps are concrete realizations of activities. Wait steps are steps that when executed can potentially wait for time, integration, manual activity, system event or other.

1. **Task step** : waiting for input from customer advisor.
2. **Rule step** : execution of **rule** machine, which potentially contains integrations and thus the possibility of delay. In the case of asynchronous integrations, a response is expected before the workflow continues.
3. **Wait step** : waiting for a specific time originator, or an input from an integration or system event.

## Processes

A series of data changes in different systems can result in a process being started. Section 2.2 Functional solution overview, provides an overview of the overall flow from initiating event to executed consequence.

The lists in the following subsections are based on events described in [Appendix 03A.8], but added together or split up where deemed appropriate in relation to the definition of workflows in the process engine.

As a starting point, each of the events below will be mapped to an independent workflow of the same name. Event variants mentioned at level 2 in the bullet points in the subsections below are likely to be realized as an initial branch in the workflow.

In connection with the later phases, a decision will be made as to whether there is a need to adapt which events are responded to.

The design of the individual processes initiated by the events is not detailed in the clarification phase, but is built up continuously in detailed design.

### Changes from gameplay data

* Changes in participants
  + player tended leave
  + player accidentally leave
  + player joins
  + player moves room
  + player kicked out
* Keywords changes
  + keyword erased
  + keyword replaced
* Time start changed
* Time end changed

### Changes from event data

* event name replaced
* date Started changed
* date ended changed
* event status changed
* Categories changed
* Labels changed

# Data

This section contains a description of what data is operated with, as well as how to read and write.

## File management

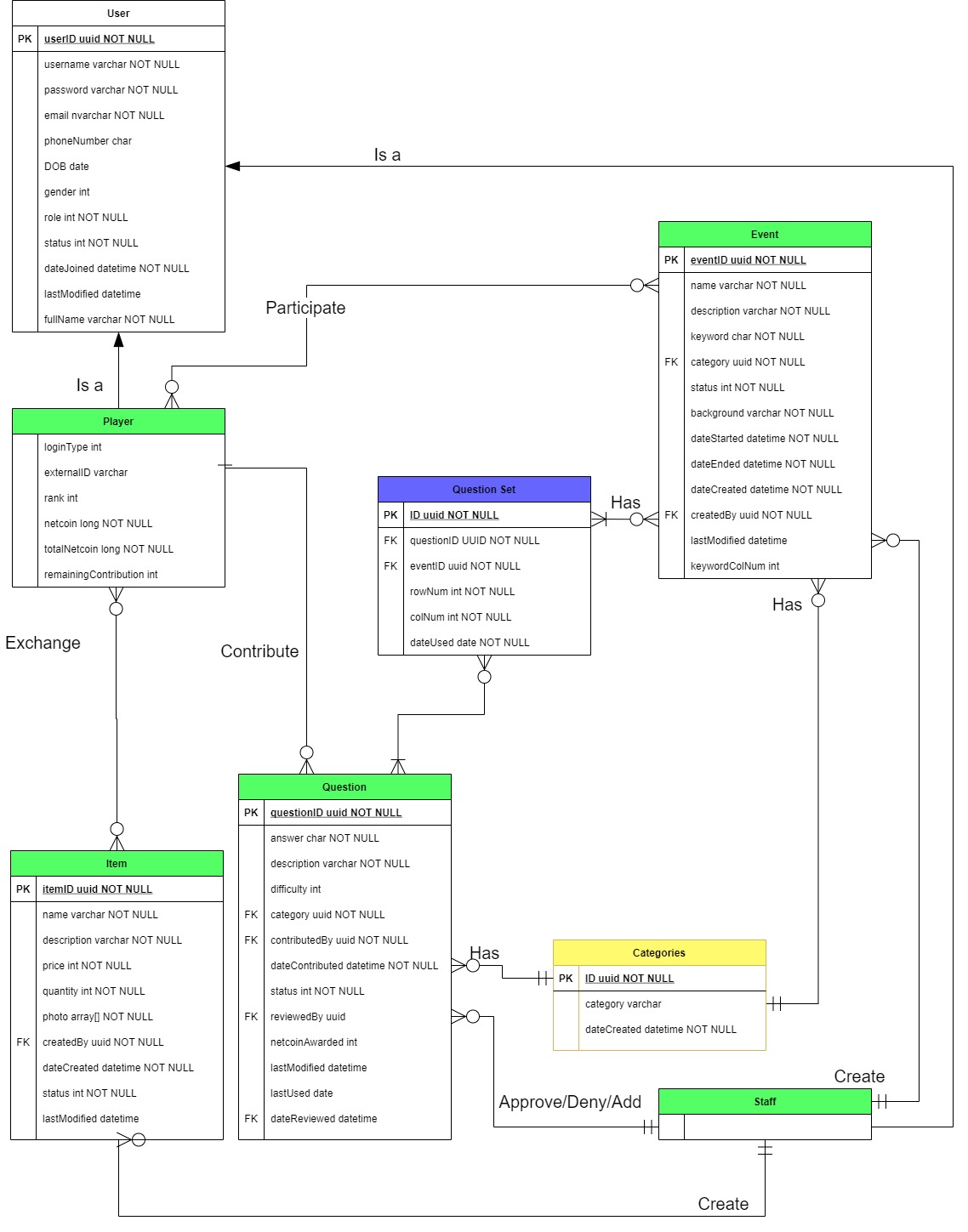
The system handles a number of files, including attachments for items and events. There are basically no restrictions in relation to allowed file types. However, since it is not appropriate to be able to persist all files regardless of size, an upper file size limit has been set for uploading files.

## Information model

The information model is basic, cross-cutting and comprehensive. Details can be found in [DD120].

## Data model for batch and operations administration

The figure below shows the data model for batch settlement and operations administration.



**Figure 6** Data model for batch settlement and operations administration

### User

|  |  |
| --- | --- |
| Attributes | Description |
| ID: uuid | Id number of an user |
| User name: varchar | The username of an user |
| full Name: varchar | The full name of an user |
| password: varchar | password of the user account |
| E-Mail: nvarchar | Email connect with the account of an user |
| Phone number:char | Phone number for an user |
| Date of birth: date | Date of birth of an user |
| Gender: int | Gender of an user |
| Role: int | The role of an user  0 -> admin 1 -> user |
| Status: int | the current status of an user  0 -> active 1 -> deactive |
| Date joined:datetime | The date and time the account of an user was created |
| Last modified:datetime | The date and time the account last changed the information |

### Player

|  |  |
| --- | --- |
| Attributes | Description |
| netcoin: long | the amount of netcoin |
| total Netcoin: long | the total amount of netcoin and user having |
| rank: int | the rank of an user in the total ranking |
| login Type: int | how the player login  0 -> by email 1 -> by social credential |
| external ID: varchar | The external ID |
| remaining Contribution: int | The question you could contributed in month |

### Event

|  |  |
| --- | --- |
| Attributes | Description |
| ID: uuid | the ID of an event |
| Name: varchar | the name of an event |
| Description: varchar | the description of an event |
| Keyword: char | the main keyword to create the puzzle of an event |
| keyword\_col\_num: int | position of the keyword in the puzzle |
| category: uuid | the category of an event |
| status: int | the current status of an event  -1 -> cancel 0 -> draft  1 -> publish 2 -> finished |
| background: varchar | the background of an event |
| date started: datetime | the date and time an event start |
| date ended: datetime | the date and time an event end |
| date created: daytime | the day and time an event created |
| created by: uuid | the ID of the creator of an event |
| last modified: datetime | the last time that the event informations are changed |

### Question

|  |  |
| --- | --- |
| Attributes | Description |
| question ID: uuid | the ID of an question |
| answer: char | the answer of an question |
| description: varchar | the description of an question |
| difficulty: int | the difficulty of an question |
| category: uuid | the category an question |
| contributed by: uuid | the ID of the user contributed an question |
| date contributed: datetime | the day and time an question contributed |
| last modified: datetime | The last time question had modified |
| last used: datetime | the last time question had used in puzzle |
| status: int | the current status of an question  -1 -> rejected  0 -> pending 1 -> approved |
| reviewed by: uuid | the ID of the staff approved an question |
| date reviewed: datetime | the date and time an contributing question reviewed |
| netcoin awarded: int | the amount of netcoin awarded to the user |

### Question Set

|  |  |
| --- | --- |
| Attributes | Description |
| ID: uuid | the ID of an question set |
| questionID: uuid | the ID of an question |
| eventID: uuid | the ID of the event |
| rowNum: int | the row order of the question in the puzzle |
| colNum: int | the column order of the question in the puzzle |
| dateUsed: date | the date that the question is used |

### Categories

|  |  |
| --- | --- |
| Attributes | Description |
| ID: uuid | the ID of the category |
| category: varchar | the category name |
| dateCreated: datetime | the date that the category been created |

### Item

|  |  |
| --- | --- |
| Attributes | Description |
| ID: uuid | the id of an item |
| Name: varchar | the name of an item |
| description: varchar | the description of an item |
| price: int | The netcoin price of an item |
| quantity: int | the amount of an item to buy |
| photo: array[] | the set of images of an item |
| created by: uuid | the id of staff added the item |
| date created: datetime | the date and time an item created |
| status: int | the current status of an item  0 -> out of stock 1 -> in stock |
| lastModified: datetime | the date that the item information last modified |

## Standard software

This section describes the standard software on which the system is built.

### IntelliJ IDEA

Version: 2020.2.3

Website: https://www.jetbrains.com/idea/

IntelliJ IDEA is an integrated development environment (IDE) written in Java for developing computer software. It is developed by JetBrains (formerly known as IntelliJ), and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition. Both can be used for commercial development.

### Java 15.0.1

Version: 15.0.1

Website: https://www.oracle.com/java/technologies/javase/15-0-1-relnotes.html

Java is the basic programming language for the system. Introduced in 1995, Java is widely used in many Enterprise solutions.

### NPM (Node Package Manager)

Version: 6X

Website: https://docs.npmjs.com/cli/v6/commands/npm-install

NPM (Node Package Manager) is a package manager for the JavaScript programming language. npm, Inc. is a subsidiary of GitHub, an American multinational corporation that provides hosting for software development and version control with the usage of Git. It is the default package manager for the JavaScript runtime environment Node.js.

### Springboot

Version: 2.4.1

Website: https://spring.io/projects/spring-boot

Spring is a frequently used framework that provides a large, coherent tool portfolio for Java applications. Spring contributes with, among other things, IoC and lays the foundation for a number of other areas, e.g. compared to JPA, batch and AOP.

### Hibernate

Version: 5.4

Website: https://hibernate.org/orm/releases/

Hibernate is an object-relational mapping (ORM) framework for Java. It handles the persistence of Java class content into relational databases. The JPA specification is used with Hibernate as provider.

### ReactJS

Version: 17.0.1

Website: https://reactjs.org/blog/2020/10/20/react-v17.html

ReactJS is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies

### PostgreSQL

Version: 13.1

Website: https://www.postgresql.org/download/

PostgreSQL is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley.

# Deployment

The family benefit solution is divided into the following components, which can be deployed independently:

1. Professional system
2. Citizen self-service
3. Batch settlement and operations administration

However, there are some significant indirect dependencies between the systems:

● All three components work on the Family Benefit database, and changes to the database may therefore require updating of all three components.

● All three components use a copy of the same performance kernel (a jar file located inside each of the three components). Each component can use different versions of the performance core, but it should be avoided that there are too many versions between e.g. the subject system and self-service, as it involves less risk of error if minor changes are tested on an ongoing basis, rather than making major upgrades with many changes.

The artifacts are built by Jenkins based on a concrete revision of the source code. Artifacts that are candidates for placement in the production environment will also be placed in the project's artifact repository, where it will have a unique version number. This means that all releases of the Family Benefit applications have an associated version number, and that the specific barley artifacts from a release can be found in the artifact repository.

In addition to an artifact, a deployment also contains a number of configurations and possibly database scripts. The same artifacts can be used for all environments, but configurations and database scripts are usually environment-specific, which is why separate configurations are used per environment. The source code itself and the artifacts do not contain any hard-coded configuration parameters - these are all placed in the external environment-specific configuration. Parameters are specified in configuration files or as system parameters in the database (see section 8.3.1).

# Performance

The system is designed to ensure good performance. This is ensured through distinct use of e.g. caching and data replication. In addition, the architecture is designed so that there is the possibility of both vertical and horizontal scaling, which helps to ensure high performance.

## Capacity

Based on the requirements of the game to make sure the services have enough spare capacity to handle any likely increases in workload, and enough buffer capacity to absorb normal workload spikes, between planning iterations and to be available for a quite amount of users to be access at the same time, but still be able to optimized the given resources and prevent wasting on those resources. Also estimated and calculated based on various types of previous events and average to maximum number of participants, the number from 200-500 end-users is decided to be the capacity of NCP.

## Usability

Since the project scope is small, usage is only for students and HR staff so the design of the interface will ensure the learnability, efficiency and satisfaction level to increase overtime.

Throughout the website, there will be hints banner and notifications banner (see section 3.2) for further user instruction, to help reach their goals more effectively and quickly.

The following activities will have instruction banners:

User (Player):

* Question contribution
* Gameplay
* Reward exchange

HR staff:

* Create a statistical component
* Approve / decline a question
* Edit / Create a particular item, for example a game event

## Security

HTTPS protects the privacy and security of your users

HTTPS prevents intruders from being able to passively listen to communications between your websites and your users. One common misconception about HTTPS is that the only websites that need HTTPS are those that handle sensitive communications

Therefore, the application will be using HTTPS instead of HTTP for data encryption and privacy. Users could be assured about their personal information and communication across the Internet.

The system will set roles for the users so just only the admin role could see the user public information. Users can not see other user data. The NCP could secure users from unauthorized access. Only the HR staff have access to the admin dashboard for management of their choice and the overview of the NCP private data.

# 

# Security architecture

This section describes the security architecture, including how to authenticate, authorization, data security, and how network and application security is handled.

## Authentication

The NCP will utilize simple authentication for the end users since NCP’s scope is small, which is also being used on the corporate website. NCP will be able to provide trust with a customer, if the user’s credential and account exist and not being deactivated in the database. In that way, the user can use their own identities stored in the database. The user will see the login screen from their own browsers. Users without a valid credential will have to be signed up.

No user will be allowed access to any data or services from NCP without being logged in. The customer’s own policies will be applied for reactivation of accounts through the standard functionality in Azure or by the NCP policies. The Azure server supports sending automated emails and blocking users based on risk analysis of the possibility that a user account has been compromised.

## Authorization

Authorization is handled in the service core on the basis of roles and rights. The attribute values ​​are mapped via a rights matrix to rights in the system. If the user has the authority to perform certain operations on behalf of some others, this is handled by the user having to make a choice about who he or she acts as. When a user performs a call against the service core, it is the data security filter in the form of the security component that ensures that business logic is performed only on data where it is corrected; both in relation to roles / functions and in relation to which specific data you have rights to.

Business users, including administrative users, will never need direct access to a data-bearing system, such as the database, to perform daily work. Configuration takes place through the same role management and security model as the normal use of the system. Thus, at no time is system access needed to perform the daily work. This helps to ensure efficient and reliable workflows and minimizes the risk of accessing data unjustifiably.

If a user attempts to access data to which he or she is not entitled according to his or her rights, a security error is thrown.

## Security Measures

The source code of the project won’t be publicly available, as it will be stored in Netcompany’s Azure DevOps repository (see sample Figure 7).

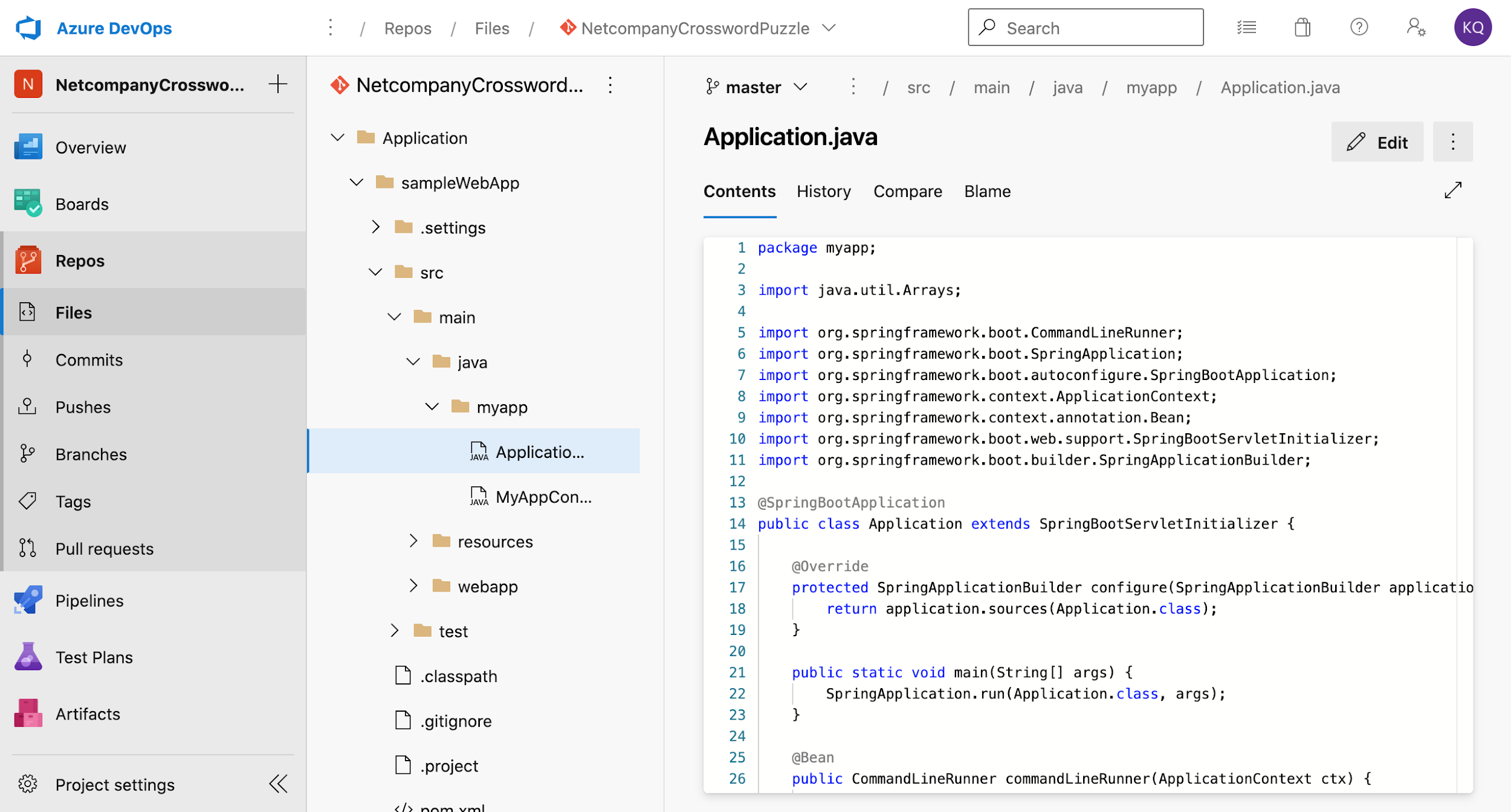


Figure 7 - Azure Devops Repository sample

The application and any associated online services should be using ‘https’ instead of ‘http' for communication, and developers should not carry any remote media or devices with unencrypted data that might compromise the integrity of NCP’s information.

Capacity management will not be implemented on the NCP sites but handled by Microsoft/Netcompany in the Azure tenant hosting the solution after the product is delivered. The system acceptance procedure will be a combination of using the release pipeline for accepting that the code is deployed to the app service running the application, and a sign-off from Netcompany based on the state of the application.

## Application security

The service core can be called via the professional system, self-service via borger.dk and via the operations administration.

Across all of these clients, there is a common way to handle authorization and authentication; via SAML 2.0 and the security component in the service core itself, which ensures that users can only access the data to which they have rights. Attempts to access data for which you do not have rights result in a rejection in the form of an error stating that you do not have the necessary rights.

The system components are developed and configured with a focus on OWASP Secure Coding Practices and are subjected to security testing / scanning. This means, among other things, that neither in the self-service solution nor in the internal systems can the system be manipulated by adjusting the URL or by constructing http POST messages yourself. All validations of both rights and parameters in relation to a given action will always be performed server-side before the action is performed.

In relation to the databases, the “least privilege” principle is used, where the individual component only has access to necessary operations on the necessary data sections (databases, tables, views). As a starting point, access is via the component's service account (without having to enter passwords in configuration files), and where there is a need to use a username / password, these will be encrypted in the configurations. In addition, access takes place through a framework with built-in validation (Hibernate), where code standards and inspections ensure that parameter validation is used and that SQL and other types of openings do not occur dynamically in relation to SQL injection.