Cahier des Charges

*Money Flop quiz*

Table des matières

[**1.** Contexte 2](#_Toc190865954)

[2. Fonctionnalités 2](#_Toc190865955)

[3. Spécifications fonctionnelles 3](#_Toc190865956)

[7. Interfaces graphiques 4](#_Toc190865957)

Groupe numéro 7 :

Théo Fillon,

Bastian Fontaine-Tranchant,

Lucas Corompt

## 1. ****Contexte****

Ce projet consiste à créer une application de jeu télévisé interactif basée sur des quiz, inspirée de *Money Drop*. L'objectif est de permettre à un joueur de répondre à une série de questions à choix multiples tout en gérant un solde d'argent qu'il peut répartir sur les différentes réponses. L'application doit être réalisée en Java et inclure deux interfaces graphiques distinctes, l'une pour l'introduction et la roue de la fortune, et l'autre pour le quiz et le calcul des mises et gains.

L'application se compose de deux grandes parties :

1. **Interface d'accueil** : Le joueur entre son pseudo et fait tourner une roue qui lui attribue une somme d'argent de départ.
2. **Interface explicative :** Le joueur prend connaissance des règles et démarre le jeu.
3. **Interface de jeu** : Le joueur répond à 10 questions avec des choix de réponses, répartit son solde d'argent sur les différentes réponses et voit son solde mis à jour après chaque question.

## ****2. Fonctionnalités****

#### **2.1 Interface d'accueil (Page d'accueil)**

* Affichage d'un message de bienvenue.
* Champ pour entrer un pseudo.
* Une roue de la fortune qui génère une somme d'argent aléatoire pour le joueur.
* Un bouton "Jouer" pour valider le pseudo et passer à l’interface explicative.

#### **2.2 Interface explicative**

* Affichage d'un message de bienvenue
* Affichage des règles du jeu
* Un bouton pour lancer le jeu

#### **2.3 Interface de jeu**

* Affichage de la première question.
* Un timer de 2 minutes qui se réinitialise à chaque question.
* 4 réponses possibles sous la question, chacune ayant un champ où le joueur peut saisir une somme d'argent à miser.
* Affichage du pseudo du joueur en haut à gauche de l'écran.
* Affichage du solde du joueur
* Un bouton "Valider" pour confirmer la réponse et la mise.
* Un indicateur du numéro de la question (1/10).
* Calcul des gains en doublant la mise sur la bonne réponse et en perdant la mise sur les mauvaises réponses.

#### **2.4 Gestion des mises et gains**

* Lors de chaque question, le joueur peut répartir son solde sur les réponses proposées.
* Si la réponse est correcte, le joueur double la somme qu'il a misée sur la bonne réponse.
* Si la réponse est incorrecte, le joueur perd l'argent misé sur les mauvaises réponses.
* Le jeu comporte 10 questions, et le but est de finir avec le plus d'argent possible.

#### **2.5 Fin du jeu**

* À la fin des 10 questions, un message d'achèvement
* Affichage de la solde final du joueur.

## ****Gestion** **des données****

#### Présentation des données utilisées

Les données utilisées dans notre application sont issues de fichiers JSON contenant des questions, leurs réponses et des anecdotes associées. Ces fichiers ont été téléchargés depuis **Open Quizz** et nous avons combiné plusieurs quiz pour créer une base de données plus variée et complète.

Le format des données est structuré en JSON, avec une organisation sous forme de liste d'objets, chaque objet représentant une question avec ses différentes réponses et une éventuelle anecdote.

#### Récupération et sauvegarde des données

Pour gérer ces données, nous utilisons une **liste Python** qui contient l’ensemble des questions chargées depuis le fichier JSON. L’application utilise la fonction **shuffle()** de la bibliothèque random pour mélanger les questions afin d’assurer une expérience variée à chaque session.

L’interaction avec les données se fait en retirant la première question de la liste après qu’elle ait été posée à l’utilisateur. Cela permet d'éviter la répétition immédiate des questions et de simuler un tirage aléatoire dynamique.

Concernant la récupération des données, notre application charge le fichier JSON au démarrage et le convertit en une liste exploitable en mémoire. Il n’y a pas de sauvegarde des données en temps réel, car les questions sont simplement utilisées en mémoire pour une session donnée.

## ****3. Spécifications fonctionnelles****

#### **3.1 Scénario de jeu**

1. **Lancement du jeu** :
   * Le joueur ouvre l'application.
   * Le pseudo du joueur est saisi.
   * La roue de la fortune tourne et attribue une somme d'argent aléatoire au joueur.
2. **Règle du jeu** :
   * Le joueur prend connaissance des règles du jeu.
   * Le joueur lance le jeu.
3. **Début de la première question** :
   * Le timer commence (2 minutes par question).
   * La question est affichée avec quatre réponses possibles.
   * Le joueur doit répartir son solde d'argent sur les réponses.
   * Le joueur valide la réponse et la mise.
4. **Après chaque question** :
   * Le solde est mis à jour en fonction de la réponse donnée.
   * Le joueur peut voir son solde, son pseudo, et l'indicateur de question actuelle (ex. 1/10).
   * Le timer se réinitialise pour la prochaine question.
5. **Fin du jeu** :
   * Après les 10 questions, un message affiche le solde final du joueur et les gains/pertes réalisés.

#### **3.2 Gestion des mises**

* Le joueur doit pouvoir saisir des montants différents pour chaque réponse.
* Le solde total doit être réparti sur les réponses choisies par le joueur.

#### **3.3 Chronomètre**

* Chaque question a un timer de 90 secondes.
* Le timer doit être réinitialisé à chaque question et visible en haut à droite de l'écran.
* À la fin du timer, si le joueur n'a pas validé sa réponse, la mise est annulée.

## ****5. Interfaces graphiques****

#### **5.1 Interface d'accueil**

* Un champ de texte pour entrer le pseudo du joueur.
* Une roue visuelle qui affiche la somme obtenue après rotation.
* Un bouton "Jouer" pour démarrer le jeu.

#### **5.2 Interface d'accueil**

* Un message de bienvenue.
* Les règles du jeu.
* Un bouton "Jouer" pour démarrer le jeu.

#### **5.3 Interface de jeu**

* Une question au centre de l'écran.
* Un timer au milieu.
* Quatre réponses sous la question avec des champs de mise.
* Le pseudo et le solde du joueur.
* Un bouton "Valider" sous les réponses pour soumettre les mises et la réponse.

#### **5.4 Utilisation de jar**