Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

SEMESTRÁLNÍ PRÁCE z UZI

Účet v bance

17. 12. 2024

MALAKHOV LEONID

1. Formulace úlohy (zadání)

Cílem projektu je vytvořit znalostní systém, který umožní uživatelům vybrat nejlepší bankovní filiálku pro založení spořicího účtu na základě zadaných podmínek. Systém porovnává nabídky tří různých bank v České republice, analyzuje zadané vstupní parametry a rozhoduje o nejvýhodnější variantě.

1.1. Specifikace

Systém poskytuje doporučení pro výběr spořicího účtu na základě:

- výše vkladu,
- počtu transakcí kartou za měsíc,
- ochoty investovat další prostředky,
- dostupnosti poboček banky v uživatelově městě nebo v nejbližším velkém městě.

Hlavní funkcionalita zahrnuje:

- **Načítání dat o bankách**: Získávání informací ze souboru `banks.json`.
- **Uživatelský vstup**: Interaktivní dotazy na základní parametry (vklad, transakce, investice, město).
- **Filtrování bank**: Výběr bank splňujících minimální požadavky (např. výši vkladu a dostupnost pobočky).
- **Rozhodování**: Výběr banky s nejvýhodnější úrokovou sazbou (základní nebo bonusovou).
- **Generování výsledků**: Poskytnutí uživateli podrobného vysvětlení důvodů doporučení.

1.2. Výchozí podmínky

Systém zpracovává data pouze pro **tři vybrané banky** v České republice.

Informace o bankách (úrokové sazby, podmínky pro bonusy, poplatky atd.) jsou pevně definovány v souboru `banks.json`.

Uživatelská data (vklad, transakce, město) jsou zadávána prostřednictvím konzolového rozhraní.

Systém předpokládá, že uživatel zadá vstupní data ve správném formátu (čísla pro vklad, transakce, investice).

1.3. Vstupní data

- **Částka vkladu (CZK)** – Minimální výše vkladu pro založení spořicího účtu.
- **Počet transakcí kartou za měsíc** – Parametr ovlivňující možnost získání bonusové úrokové sazby.
- **Ochota investovat (CZK)** – Měsíční částka, kterou je uživatel ochoten investovat.
- **Dostupnost pobočky banky**:

- Má-li uživatel zájem o bankovní pobočku ve svém městě.
- Pokud pobočka není ve městě, zadává nejbližší větší město.

1.4. Požadavky na výstupy

Výstupem systému je:

- Název doporučené banky.
- Informace o úrokové sazbě (základní nebo bonusové).
- Důvody, proč byla banka vybrána:
    - Splnění podmínek pro bonusovou sazbu (vklad, transakce, investice).
    - Výše poplatků za vedení účtu.
    - Dostupnost pobočky ve zvoleném městě.
- Detailní údaje o účtu:
    - Minimální a maximální výše vkladu.
    - Poplatky za vedení účtu.

2. Analýza úlohy

Pro vytvoření znalostního systému bylo nutné analyzovat možné způsoby řešení problému výběru banky a rozhodnout o nejvhodnější variantě. Hlavním cílem bylo navrhnout efektivní a snadno rozšiřitelný systém, který by splňoval zadání kurzu a zároveň byl uživatelsky přívětivý.

2.1. Rozbor možných postupů

Při návrhu systému byly zvažovány následující přístupy:

- **Statický výběr založený na předdefinovaných podmínkách**
    - Každá banka by měla pevně přiřazenou váhu na základě předem určených kritérií (např. úroková sazba, poplatky).
    - Nevýhoda: Tento přístup by nebyl flexibilní, protože ignoruje variabilní vstupy uživatele (výše vkladu, transakce, investice).
- **Podmínkové filtrování s následným hodnocením**
    - Systém by nejprve filtroval banky na základě základních kritérií (např. minimální vklad), a poté hodnotil zbylé možnosti podle dalších podmínek (bonusová sazba, poplatky).
    - Výhody:
        - Zajišťuje efektivní zpracování vstupů uživatele.
        - Umožňuje snadné přidání nových kritérií do filtrace.

    - Nevýhoda: Vyžaduje více modulárních kroků, což zvyšuje složitost implementace.

- **Použití expertního systému s pravidly (rule-based approach)**
    - Pravidlový systém, který na základě podmínek v databázi znalostí vyhodnocuje všechny možné kombinace.
    - Výhody: Flexibilita a možnost zahrnout složitější logiku rozhodování.
    - Nevýhody: Náročnost implementace při velkém množství bank a kritérií. Pro potřeby jednoduchého systému bylo toto řešení příliš robustní.
- **Machine Learning (strojové učení)**
    - Systém by se učil na historických datech a doporučoval nejlepší banku podle uživatelských vstupů.
    - Nevýhody:
        - Vyžaduje dostatek trénovacích dat, která nejsou v rámci projektu dostupná.
        - Pro řešení této úlohy je tento přístup zbytečně složitý.

2.2. Výběr nejvhodnější alternativy

Na základě analýzy možných postupů byl vybrán přístup **podmínkového filtrování s následným hodnocením**. Tento přístup byl zvolen z následujících důvodů:

- **Modularita**: Implementace je rozdělena do samostatných modulů, které se snadno udržují a rozšiřují. Například:
    - Modul `bank_filter.py` odpovídá za filtrování bank na základě základních podmínek (minimální vklad, dostupnost pobočky).
    - Modul `bonus_checker.py` ověřuje dodatečné podmínky pro získání bonusové sazby.
    - Modul `decision_maker.py` porovnává banky a vybírá tu nejvýhodnější.
- **Efektivní zpracování**: Systém nejprve redukuje množinu bank na základě základních kritérií, a teprve poté provádí podrobnější analýzu. To snižuje výpočetní náročnost.
- **Flexibilita**: Přidání nových kritérií (např. další bonusové podmínky) nebo bank je snadné díky strukturovaným vstupním datům (`banks.json`).
- **Jednoduchost použití**: Uživatelské rozhraní je intuitivní a využívá konzolového vstupu pro získání dat od uživatele.
- Zvolený přístup tedy splňuje všechny požadavky zadání: je dostatečně flexibilní, efektivní a realizovatelný v rámci kurzu.

3. Popis algoritmu řešení

Řešení úlohy je založeno na modularitě, kde každý modul v programu odpovídá za specifickou část funkcionality. Tato dekompozice zajišťuje přehlednost, snadnou údržbu a rozšiřitelnost systému.

Hlavní kroky algoritmu:

1. **Načtení dat o bankách**
   Data o bankách jsou uložena ve formátu **JSON** (soubor `banks.json`).
   Modul: **`data_loader.py`**

2. **Získání vstupu od uživatele**
   Program získá potřebné informace od uživatele:
   - Částka vkladu (deposit)
   - Počet transakcí kartou měsíčně (transactions)
   - Měsíční investice (investment)
   - Informace o důležitosti pobočky v daném nebo nejbližším městě (city_info)
     Modul: **`user_input.py`**

3. **Filtrování bank podle základních podmínek**
   Banky se nejprve filtrují na základě následujících kritérií:
   - Minimální a maximální povolená výše vkladu.
   - Dostupnost pobočky v uživatelem určeném městě nebo jeho okolí (pokud je požadováno).
     Modul: **`bank_filter.py`**

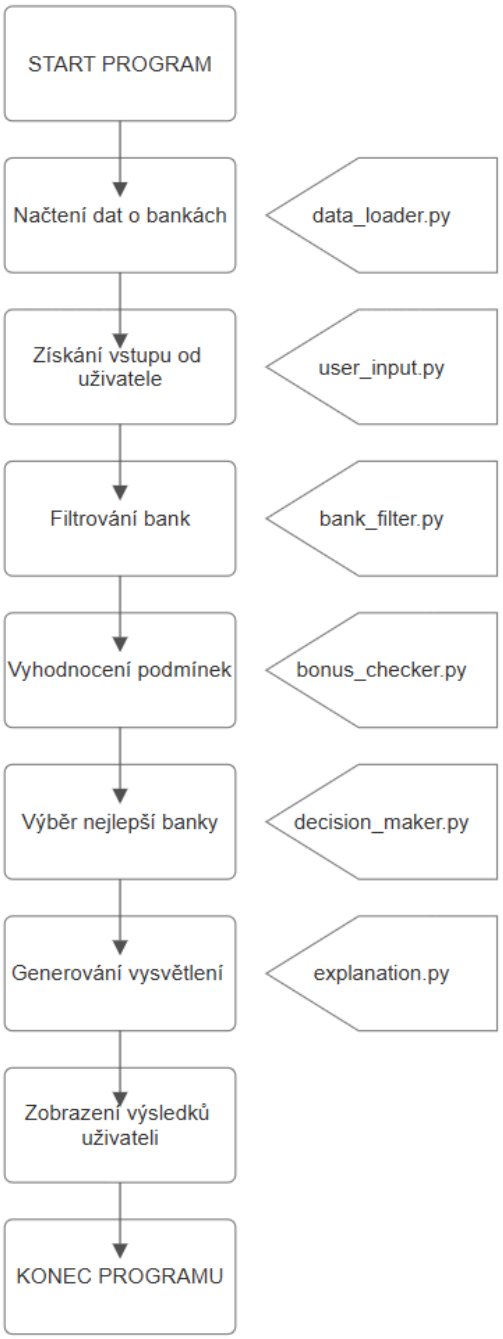
4. **Vyhodnocení bonusových podmínek a výběr nejlepší banky**
   - Pro každou banku se ověří, zda splňuje podmínky pro získání **bonusové úrokové sazby**.
   - Vyhodnocuje se:
     - Minimální vklad.
     - Minimální počet transakcí kartou.
     - Měsíční investiční závazek.
   - Pokud jsou bonusové podmínky splněny, banka získá vyšší sazbu. Jinak se uplatní základní sazba.
   - Program vybere banku s **nejvyšší dostupnou úrokovou sazbou**.
     Moduly: **`decision_maker.py`**, **`bonus_checker.py`**

5. **Vygenerování výsledného vysvětlení**
   Na základě vybrané banky a úrokové sazby program sestaví výstupní zprávu, která obsahuje:
   - Název banky.

- Důvod výběru (bonusová sazba splněna/nesplněna).
- Další podmínky účtu (např. poplatky, informace o pobočkách).
  Modul: `explanation.py`

6. **Zobrazení výsledků uživateli**
   Program zobrazí finální doporučení spolu s vysvětlením a čeká na ukončení uživatelem.
   Hlavní řídicí modul: `main.py`

Schématické znázornění návazností dílčích kroků algoritmu
Níže je uvedeno **blokové schéma**, které znázorňuje návaznosti jednotlivých modulů a kroků programu:



Obr. 1 Blokové schéma

4. Popis programu

4.1. Volba programovacího jazyka a technologie

Pro implementaci programu byl zvolen jazyk **Python**. Tento jazyk byl vybrán z následujících důvodů:

- **Snadnost použití a čitelnost** – Python má čistou a jednoduchou syntaxi, což umožňuje rychlou implementaci a snadnou údržbu kódu. To je zvlášť užitečné pro rychlý vývoj prototypů a pro programy, kde je důležitá modularita.
- **Bohatá knihovna standardních modulů** – Python nabízí širokou škálu knihoven pro práci s daty (např. `json`, pro práci s JSON soubory) a textem, což značně usnadňuje implementaci.
- **Vysoká produktivita vývoje** – Python umožňuje rychlou implementaci i testování, což je ideální pro tento projekt.
- **Kompatibilita s různými platformami** – Python je multiplatformní jazyk, což zajišťuje snadnou přenositelnost programu na různé operační systémy bez nutnosti dalších úprav kódu.

4.2. Volba datových typů a struktur

V programu byly použity následující datové typy a struktury:

- **Slovníky (`dict`)**
  Slovníky jsou použity pro reprezentaci bank, protože umožňují snadné přiřazení a hledání hodnot na základě klíčů (např. `bank["interest_rate_basic"]`). Tento typ struktury je ideální pro uchování informací o jednotlivých bankách, jako jsou úrokové sazby, poplatky, nebo bonusové podmínky.
- **Seznamy (`list`)**
  Seznamy jsou využity pro uchování seznamu bank nebo pro filtrování vhodných bank. Seznam je vhodný, protože umožňuje dynamické přidávání a mazání prvků během běhu programu.

4.3. Blokové schéma programových kroků

Návaznost mezi jednotlivými programovými moduly a kroky byla již dříve popsána v **části 3** (Popis algoritmu řešení). Tento krok je tedy již součástí popisu dekompozice algoritmu.

4.4. Popis významu použitých symbolů

Proměnné:

- `deposit` - Částka, kterou chce uživatel vložit.
- `transactions` - Počet transakcí provedených uživatelem.
- `investment` - Měsíční investice, pokud existuje.
- `city_info` - Informace o požadavku na pobočku banky.
- `banks` - Seznam bankových dat načtený ze souboru.
- `eligible_banks` - Seznam bank, které splňují podmínky uživatele.

Funkce:

- `choose_best_bank()` - Funkce pro výběr banky s nejvyšší úrokovou sazbou.

- `get_user_input()` - Funkce pro získání vstupních dat od uživatele.
- `filter_banks()` - Funkce pro filtrování bank podle základních podmínek.
- `check_bonus_conditions()` - Funkce pro ověření, zda banka splňuje podmínky pro bonusovou sazbu.

4.5. Výčet specifických vlastností programových modulů

- **`data_loader.py`**:
  - Načítá bankovní data z externího souboru (JSON), což umožňuje snadnou úpravu dat bez změn v kódu.
- **`user_input.py`**:
  - Zaručuje, že uživatel zadá pouze validní hodnoty (čísla pro vklad, počet transakcí atd.).
  - Umožňuje flexibilitu pro volitelné informace o pobočkách.
- **`bank_filter.py`**:
  - Modul umožňuje filtrování bank jak podle základních podmínek, tak i podle dostupnosti poboček v konkrétním městě.
- **`bonus_checker.py`**:
  - Tento modul obsahuje logiku pro zjištění, zda banka splňuje podmínky pro bonusovou úrokovou sazbu.
- **`decision_maker.py`**:
  - Odpovědný za výběr nejlepší banky, přičemž zohledňuje bonusové a základní sazby.
- **`explanation.py`**:
  - Modul generuje podrobné vysvětlení výběru banky a zobrazuje důvody výběru uživateli.

5. Popis obsluhy programu

5.1. Instalace programu

Pro úspěšnou instalaci a běh programu na vašem zařízení jsou požadovány následující kroky:

- **Požadavky na software**:
- **Python 3.x** (doporučeno verze 3.7 nebo novější).
- Knihovna **JSON** pro načítání a zápis dat (standardní knihovna Pythonu).
- Pokud máte nainstalován Python, není potřeba instalovat žádné další knihovny.
- **Postup instalace**:
- **Stáhněte si kód programu**: Program je dostupný v repozitáři GitHub. Stáhněte si celý adresář s kódem nebo získejte soubor jako ZIP.
- **Ověřte instalaci Pythonu**: Spusťte příkaz:
  `python --version`
  - Tento příkaz by měl vrátit verzi Pythonu, která je nainstalována na vašem systému.

- **Stáhněte si soubor s daty o bankách (JSON)**: Soubor s daty je nezbytný pro správnou funkci programu. Můžete jej stáhnout z odkazu (pokud je poskytnut) nebo použít vlastní soubor.
- **Spusťte program**: Po stažení a přípravě datového souboru přejděte do složky, kde je kód uložen, a spusťte hlavní skript:
  `python main.py`

5.2. Spuštění programu

Po úspěšné instalaci a nastavení programu můžete spustit aplikaci podle následujících pokynů.

- **Spuštění programu z terminálu**: Program je možné spustit přímo z příkazového řádku nebo terminálu pomocí Pythonu. Stačí přejít do adresáře, kde je uložen soubor `main.py`, a spustit:
  `python main.py`

Interaktivní uživatelské rozhraní: Po spuštění programu se zobrazí textová výzva, která požaduje zadání údajů. Program bude požadovat následující vstupy:

- **Vklad**: Uživatel zadá částku, kterou chce vložit na účet (např. 5000 Kč).
- **Počet transakcí**: Uživatel zadá počet transakcí, které provedl (např. 10).
- **Měsíční investice**: Pokud má uživatel závazek k pravidelným investicím, zadá částku (např. 1000 Kč). Pokud nemá investice, může zadat hodnotu `0`.
- **Informace o městě**: Uživatel zadá město, ve kterém hledá banku (pokud je to relevantní) nebo může tento údaj vynechat, pokud není požadováno.

Po zadání těchto hodnot program provede filtrování bank a vybere tu, která nabízí nejvýhodnější úrokovou sazbu podle zadaných podmínek.

5.3. Podmínky pro zadávání vstupních dat

Aby program správně fungoval, je nutné dodržovat následující podmínky při zadávání vstupních dat:

- **Vklad** (numeric):
  - Vklad musí být číselná hodnota, která může obsahovat desetinná místa (např. 5000.75).
  - Hodnota vkladu musí být alespoň tak velká jako minimální požadovaný vklad banky (dle filtrů).
- **Počet transakcí** (integer):
  - Počet transakcí musí být celé číslo.
  - Program zkontroluje, zda tento počet transakcí odpovídá podmínkám pro bonusovou úrokovou sazbu.
- **Měsíční investice** (numeric):
  - Měsíční investice musí být číselná hodnota.

    - Pokud uživatel nemá investice, může zadat `0` nebo ponechat tuto hodnotu prázdnou, pokud není povinná.

- **Město** (string):
    - Uživatel může zadat název města, pokud chce filtrovat banky podle dostupnosti poboček.
    - Město by mělo být zadáno přesně, jak je uvedeno v databázi, jinak nebude možné najít odpovídající pobočku.

5.4. Použití programu

Program umožňuje uživateli získat přehled o nejvýhodnější bance pro otevření spořicího účtu na základě zadání následujících parametrů:

- Výše vkladu.
- Počet provedených transakcí.
- Výše měsíční investice (pokud existuje).
- Možná požadavky na dostupnost pobočky v konkrétním městě.

Po zadání všech potřebných údajů program:

- Načte data o bankách.
- Filtrování bank podle základních podmínek (minimální vklad, maximální vklad, atd.).
- Zkontroluje, zda uživatel splňuje podmínky pro bonusové úrokové sazby.
- Vybere banku s nejvýhodnější úrokovou sazbou.
- Vygeneruje textové vysvětlení výběru banky.

5.5. Ukončení programu

Po dokončení výběru nejlepší banky a zobrazení výsledku je možné program ukončit stisknutím klávesy `Enter`. Program následně zavře všechny otevřené soubory a ukončí běh.

Pokud chcete spustit program znovu nebo provést jiné výpočty, jednoduše opakujte postup spuštění programu.

6. Rozbor výsledků, zhodnocení

6.1. Rozbor dosažených výsledků

Program, který jsem vyvinul, splňuje požadavky zadání, tedy vytváří základní znalostní systém pro výběr nejlepší banky pro otevření spořicího účtu na základě individuálních požadavků uživatele. Tento systém hodnotí banky podle několika kritérií, jako jsou:

- Minimální a maximální vklad.
- Počet měsíčních transakcí a splnění podmínek pro bonusovou úrokovou sazbu.
- Měsíční investice.
- Geografická dostupnost poboček v určitém městě.

Program dokáže:

- Načíst potřebná data o bankách.
- Filtrovat banky podle zadaných vstupních parametrů.
- Vybrat banku s nejvýhodnější úrokovou sazbou na základě zadaných podmínek.
- Vygenerovat textové vysvětlení pro uživatele, které informuje o tom, proč byla tato banka vybrána.

6.2. Vyzdvihnutí kladů a přínosu prezentovaného řešení

- **Srozumitelnost a přehlednost**: Program je navržen tak, aby byl co nejvíce srozumitelný a snadno ovladatelný. Uživatelé mohou snadno zadávat vstupní údaje a získat rychlou zpětnou vazbu o nejlepší bance.
- **Modularita a flexibilita**: Program je rozdělen do několika modulů (např. `bank_filter`, `decision_maker`, `user_input`), což usnadňuje jeho údržbu, rozšíření a testování. Modularita také znamená, že jednotlivé části programu mohou být snadno upravovány nebo vyměňovány bez zásadního ovlivnění ostatních částí systému.
- **Rozšiřitelnost**: Program je navržen tak, aby mohl snadno zahrnovat nové banky a parametry (např. nové podmínky pro bonusovou sazbu, další kritéria pro výběr banky). Lze přidávat nové funkce, aniž by bylo nutné zásadně měnit stávající kód.
- **Uživatelská přívětivost**: Program poskytuje jasné instrukce pro zadávání vstupních dat a správně ošetřuje chybné vstupy (např. pokud uživatel zadá neplatnou hodnotu, program se pokusí získat platný vstup).
- **Přesnost výběru banky**: Program na základě zadaných vstupních dat dokáže správně vybrat banku, která je pro uživatele nejvýhodnější. Filtrování bank je efektivní a na základě zadaných parametrů je vždy vybrána banka, která splňuje požadavky.

6.3. Kritika nedostatků

I přesto, že program splňuje požadavky a je funkční, existují určité oblasti, které by bylo možné zlepšit nebo vylepšit:

- **Omezené testování**: Program byl testován s několika standardními vstupy, ale nebyl dostatečně testován na okrajových případech, jako jsou neobvyklé hodnoty vkladů nebo extrémní hodnoty počtu transakcí. V budoucnu by bylo dobré provést komplexnější testování.
- **Geografická dostupnost poboček**: Program zohledňuje pouze přítomnost poboček v uživatelem zadaných městech. V případě většího městského aglomerátu nebo regionálních požadavků by bylo vhodné zlepšit tento mechanismus o možnost vyhledávání podle širšího geografického rozmezí nebo využít pokročilejší API pro mapování poboček.
- **Možnost zahrnutí více bank**: Program v současnosti pracuje s jedním souborem dat o bankách. V budoucnu by bylo možné integrovat více zdrojů dat (např. připojení k veřejným databázím bank) pro zajištění aktuálnosti informací.
- **Uživatelský zážitek**: I když je program funkční, interaktivní uživatelské rozhraní (např. grafické rozhraní místo textového vstupu) by mohlo přispět k lepšímu uživatelskému zážitku, zejména pro uživatele, kteří nejsou zvyklí pracovat s příkazovým řádkem.
- **Zpracování neúplných vstupních dat**: Pokud uživatel nezadá všechny požadované údaje (např. nezadá město, pokud je požadováno), program pokračuje dál. V budoucnu by bylo vhodné přidat možnost upřesnit požadované údaje nebo nabídnout alternativy.

6.4. Možnosti zlepšení

- **Integrace s externími API**: Pro získávání nejaktuálnějších dat o bankách by bylo užitečné propojit systém s veřejnými API, která poskytují informace o bankách a jejich podmínkách.
- **Podrobnější analýza bonusových podmínek**: Mohlo by být užitečné přidat více parametrů pro analýzu bonusových podmínek bank, např. zohlednění dalších faktorů jako výše poplatků za účet nebo dostupnost dalších produktů (úvěry, pojištění, atd.).
- **Přidání vizualizace**: Pro lepší přehlednost a porovnání různých bank by bylo možné přidat grafické zobrazení (např. grafy nebo tabulky) pro výběr nejvhodnější banky.

7. Závěr

Celkově řešení splňuje všechny požadavky zadání a poskytuje funkční nástroj pro výběr nejlepší banky na základě individuálních potřeb uživatele. Systém je efektivní, přehledný a snadno udržovatelný, což z něj činí solidní základ pro budoucí rozšíření. Možnosti dalšího rozvoje zahrnují integraci s externími daty, vylepšení uživatelského rozhraní a zlepšení analýzy podmínek bank, což může ještě více zvýšit hodnotu a přesnost systému.