Online-Marktplatz: Projektbeschreibung und Aufgabenstellung

Projektbeschreibung

Der Online-Marktplatz, den Sie entwickeln werden, soll eine voll funktionsfähige Plattform zum Kaufen und Verkaufen von Artikeln bieten. Nutzer können Artikel einstellen, auf Artikel bieten, Artikel kaufen und verkaufen sowie mit anderen Nutzern interagieren. Die Plattform bietet eine Vielzahl von Funktionen, um den Kauf- und Verkaufsprozess zu erleichtern und die Nutzererfahrung zu verbessern.

Features des Marktplatzes

- **Benutzerverwaltung**: Registrierung, Login, Logout, Profilverwaltung.
- **Freundesverwaltung**: Freunde hinzufügen und entfernen, Anzeige gemeinsamer Freunde.
- **Artikelverwaltung**: Artikel hinzufügen, löschen, suchen und anzeigen.
- **Auktionsverwaltung**: Bieten auf Artikel, Anzeige von Geboten, Anzeige gewonnener und verkaufter Artikel.
- **Empfehlungssystem**: Anzeige empfohlener Artikel basierend auf dem Benutzerverhalten.

Aufgabenstellung

Das Projekt ist in drei Teilprojekte unterteilt, wobei Sie eine bereits implementierte Rohversion des Marktplatzes in Python erhalten. Zu den drei Praktikumsterminen müssen Sie das jeweilige Teilprojekt in einer Powerpoint Präsentation vorstellen.

Das Rohprojekt besteht aus den folgenden Dateien:

Dateien, die Sie bearbeiten werden:	
`praktikumsgruppen.py`	Sortiert Nutzer in Praktikumsgruppen ein. Editieren Sie im 1. und 3. Praktikum.
`max_heap.py`	Speichert die Auktionen in einem Heap. Editieren Sie im 2. Praktikum.
`users.py`	Definiert einen Container für alle Nutzer der Plattform. Editieren Sie im 3. Praktikum
`auctions.py`	Definiert Container für alle Auktionen. Editieren Sie im 2. Praktikum.
`transactions.py`	Speichert die getätigten Transaktionen in eine Hashtabelle. Editieren Sie im 2. Praktikum.
Dateien, die Sie sich vielleicht ansehen möchten:	
`item.py`	Definiert ein Produkt, das ersteigert werden kann
`stack.py`	Implementierung eines Stacks
`auction.py`	Definiert eine Auktion
`user.py`	Definiert einen Nutzer des Marktplatzes
`systemmessages.py`	Anzeige von Systemnachrichten auf der GUI
`trie.py`	Implementierung der Datenstruktur Trie
`avl_tree.py`	Implementierung eines AVL-Baums
Unterstützende Dateien, welche Sie ignorieren können:	
`gui_marketplace.py`	Enthält die main. Mit „python gui_marketplace.py“ starten Sie die Anwendung
`auctionapp_init.py`	Initialisiert die GUI
`simulator.py`	Simuliert Kauf- und Verkauf-Vorgänge auf dem Marktplatz

Teilprojekt 1: Projektanalyse und Verständnis

Ziele:

- Vertraut machen mit dem bestehenden Code.
- Beantwortung grundlegender Fragen zum Code und den verwendeten Datenstrukturen.

Aufgaben:

1. **Code-Analyse:**
   - Laden Sie die bereitgestellte Rohversion des Online-Marktplatzes herunter, starten Sie das Programm mit „python gui_marketplace.py“ und testen Sie das Programm. Ihr Nutzername ist Ihre GM-ID und das Passwort ist „abcde“.
   - Machen Sie sich mit der Struktur und den Hauptkomponenten des Projekts vertraut. Schauen Sie sich dafür insbesondere diese Dateien an:
     1. user.py, praktikumsgruppen.py, users.py
     2. auction.py, auctions.py, item.py, stack.py
     3. systemmessages.py, trie.py
2. **Analyse der ADTs und Datenstrukturen:**
   - In welcher Datenstruktur sind die Nutzer des Marktplatzes gespeichert? Schauen Sie sich dafür die users.py und die praktikumsgruppen.py an. Erweitern Sie die Klasse SetNode in praktikumsgruppen.py um die private Membervariable „_praktikumsgruppe“ und implementieren Sie die Methoden „create_groups()“ und „get_groupmembers()“ in praktikumsgruppen.py.
   - Welche Laufzeit hat aufgrund der gewählten Datenstruktur ein Zugriff auf einen bestimmten Nutzer des Marktplatzes? Welche Laufzeit hat die Methode „get_groupmembers()“ in Abhängigkeit der Anzahl Nutzer des Marktplatzes?
   - In welchem ADT werden alle Auktionen gespeichert und als welche Datenstruktur wird er implementiert? (s. auctions.py)
   - Welcher ADT wird in der Klasse systemmessages.py genutzt und warum? Als welche Datenstruktur wird der ADT implementiert?
   - Wie wird gespeichert in welcher Reihenfolge die Gebote auf eine Auktion abgegeben wurden? Wie erhält man die zuletzt abgegebene Auktion? (s. auction.py)
   - Wenn Sie nach einem Produkt suchen, werden Ihnen in einem Tooltip Autoergänzungsvorschläge angezeigt auf die Sie klicken können. Die Datenstruktur dahinter ist ein Trie, s. trie.py. Informieren Sie sich über Tries (s. Vorlesungsfolien). Erweitern Sie den Trie so, dass Sie diesen auch nutzen können, um im Suchfeld nach Nutzer-IDs zu suchen. S. auctionapp_init.py: initialize_trie().
   - Diese Autoergänzung kann auch durch einen AVL-Baum implementiert werden. Schauen Sie sich die Klasse AVLTree in avl_tree.py an und dort insbesondere die Methode find_most_likely_words(). Ändern Sie die Methode show_suggestions() in auctionapp_init.py, sodass die Vorschläge vom AVL-Baum genutzt werden. Können Sie Unterschiede in der Laufzeit zwischen der Trie und der AVL-Baum Implementierung feststellen?
3. **Hashfunktion verbessern:**
   - Transaktionen (Verkäufer, Käufer, Preis, Zeitstempel, Produktname) werden in einer eigenen Hashtabelle gespeichert. Schauen sie sich die implementierte Hashtabelle in transactions.py mit ihrer Hashfunktion an. Wie funktioniert die Hashfunktion? Was passiert bei einer Kollision?
   - Aktuell sorgt die Hashfunktion für viele Kollisionen. Verbessern sie die Hashfunktion in der Klasse Transactions (s. transactions.py) so das deutlich weniger Kollisionen

passieren. Die Kollisionen werden alle 30 Sekunden über die Systemnachrichten mitgeteilt.

4. **Dokumentation**
   - Erstellen Sie eine kurze Präsentation, in der die oben gestellten Fragen beantwortet werden.
   - Optional: Fügen Sie Kommentare im Code hinzu, um den Zweck und die Funktionsweise der Hauptfunktionen zu erklären.

Teilprojekt 2: Erweiterung der Artikel- und Auktionsverwaltung

Ziele:

- Implementierung zusätzlicher Funktionen für die Artikel- und Auktionsverwaltung.
- Verbesserung der Benutzerfreundlichkeit und Funktionalität.

Aufgaben:

1. **Analyse der ADTs und Datenstrukturen:**
    - In welcher Datenstruktur werden alle Gebote auf eine Auktion gespeichert? Warum wird diese Datenstruktur genutzt? (s. auction.py)
    - Lassen Sie die aktivste Auktion, d.h. die auf die die meisten Nutzer bieten, als Systemnachricht anzeigen. Nutzen Sie eine geeignete Datenstruktur, die die Auktionen so anordnet, dass die aktivste Auktion in konstanter Zeit abrufbar ist (ein Template ist Ihnen in der Datei max_heap.py gegeben). Prüfen Sie alle 30 Sekunden ob sich die aktivste Auktion geändert hat und informieren Sie den Nutzer wieder über die Systemnachricht (diese Funktionalität existiert bereits: s. update_listboxes() in gui_marketplace.py). Die Schwierigkeit ist hier, dass sich die Anzahl der Gebote pro Auktion andauernd ändert und sich damit der Ort an dem die Auktion in der Datenstruktur gespeichert wird ebenfalls ändern muss/kann. Sie müssen deshalb zusätzlich eine Hashtabelle nutzen, die den Ort speichert an dem die Auktion in der eigentlichen Datenstruktur gespeichert ist (s. max_heap.py für Details). Es sind **ALLE** Hilfsmittel zugelassen.
    - Vergleichen Sie die Laufzeit vom Max-Heap mit und ohne die Hashtabelle. Wann lohnt es sich die Hashtabelle mit dem Max-Heap zu benutzen? Wann sollte man nur einen Max-Heap benutzen?
2. **Interaktion und Feedback:**
    - Fügen Sie eine Bewertungsfunktion (1 bis 5 Sterne) hinzu, mit der Nutzer Verkäufer bewerten können. Der am besten bewertete Verkäufer soll alle 30 Sekunden als Systemnachricht angezeigt werden. Es soll die am besten geeignete Datenstruktur gewählt werden. Beachten Sie, dass sich der am besten bewertete Verkäufer während der Laufzeit des Programms verändern kann (s. Ähnlichkeit zu aktivste Auktion oben). In auctions.py finden Sie bereits die Methode get_top_rated_user(), die den beliebtesten Nutzer liefert. Dieser wird auch bereits als Systemnachricht ausgegeben, s. update_listboxes() in gui_marketplace.py. Sie sollen lediglich eine bessere Datenstruktur wählen und die Methode get_top_rated_user() anpassen. S. TODOs in auctions.py.

Erstellen Sie eine kurze Präsentation, in der Sie die oben gestellten Fragen beantworten und Ihren Code präsentieren.

Teilprojekt 3: Freundesverwaltung, Gebotsagenten und Empfehlungssystem

Ziele:

- Implementierung und Verbesserung der Freundesverwaltung.
- Einführung von intelligenten Gebotsagenten, die strategisch und optimiert bieten.
- Entwicklung eines Empfehlungssystems basierend auf dem Nutzerverhalten.

Aufgaben:

1. **Freundesverwaltung:**
   - Implementieren Sie eine Funktion, die dem Nutzer solche Nutzer als Freunde vorschlägt, die mit möglichst vielen seiner Freunde befreundet sind (s. users.py, suggest_friends()).
   - Schreiben Sie eine Funktion, die prüft ob zwei Nutzer irgendwie über deren Freundesnetzwerk (ersten, zweiten, dritten Grades) verbunden sind (s. users.py, are_users_connected()).
   - Wie Sie evtl. bereits gesehen haben, haben wir für jeden Nutzer einen zufälligen Wohnort in NRW (oder in der Nähe) generiert. Berechnen Sie den Abstand von dem Wohnort Ihres users zu dem Wohnort aller anderen Nutzer, mit denen Sie über mehrere Ecken befreundet sind, um sich weitere Nutzer als potenzielle Freunde empfehlen zu lassen. Der Abstand soll der Abstand sein, den Sie mit einem Fahrzeug zwischen beiden Wohnorten zurück legen müssen. Nutzen Sie dafür die Bibliothek: https://osmnx.readthedocs.io/en/stable/
   - Hier ist ein bisschen Code für osmnx:

```
import osmnx as ox
map_graph = ox.graph.graph_from_address('Gummersbach, Steinmüllerallee
1, Germany', dist=5000, network_type='bike')
origin_point = (50.985108, 7.542490)   # Breiten- und Längengrad
destination_point = (51.022255, 7.562705)
origin = ox.distance.nearest_nodes(map_graph, origin_point[1],
origin_point[0])
destination = ox.distance.nearest_nodes(map_graph,
destination_point[1], destination_point[0])
shortest_path = ox.distance.shortest_path(map_graph, origin,
destination, weight='length')
```

2. **Portoberechnung (optionale Aufgabe):**
   - Das Porto für den Versand der Artikel, sei proportional zu der Distanz zwischen Verkäufer und Käufer. Berechnen Sie für die Auktionen, auf die der Nutzer gerade bietet, das Porto (0,1 € pro 5 km) und geben Sie das Porto bei der Anzeige der Auktionen, auf die der Nutzer gerade bietet, an.
3. **Verwaltung der Studierenden in Praktikumsgruppen:**
   - Schauen Sie sich die Datei praktikumsgruppen.py an und implementieren Sie die Klasse Praktikumsgruppen als Menge von disjunkten Mengen. Nutzen Sie weighted Quick-Union mit Pfadverkürzung.
   - Bei dieser Implementierung können Sie die SetNode Klasse nutzen. In dieser können Sie die in Praktikum 1 hinzugefügte Membervariable _praktikumsgruppe wieder auskommentieren. Sie müssen auch die Methoden „create_groups()“ und „get_groupmembers()“ in praktikumsgruppen.py neu implementieren.
   - Erläutern Sie wie zwei Praktikumsgruppen zusammengelegt werden können. Was passiert in der Menge der disjunkten Menge?

Erstellen Sie eine kurze Präsentation, in der Sie die oben gestellten Fragen beantworten und Ihren Code präsentieren.

Abschluss

Nach Abschluss der drei Teilprojekte sollten Sie eine umfassende, funktionsfähige und benutzerfreundliche Online-Marktplatz-Plattform entwickelt haben. Durch diese Aufteilung in drei Teilprojekte können Sie schrittweise die Komplexität des Projekts bewältigen und ihre Kenntnisse in Datenstrukturen und Algorithmen vertiefen. Jedes Teilprojekt baut auf dem vorherigen auf und führt neue Konzepte ein, die für die erfolgreiche Implementierung des gesamten Systems notwendig sind.

Viel Erfolg!