Bewertungskriterien für den dritten Anwendungsfall

Bewertung

Softwaredesign:

	CodeFuse-DeepSeek-33B	**CodeQwen1.5-7B-Chat**	**Nxcode-CQ7B-orpo**	**OpenCodeInterpreter DS-33B**	**Phind-Codellama-34b-v2.0**	**Summe:**
Lauffähig	**3**	**1**	**2**	**2**	**2**	**10**
Laufzeit aggregiert	**4947s → 82,45min → 1,37h**	**2464s → 41,06min**	**9009s → 150min → 2,5h**	**19292s →321,53min. → 5,35h**	**1910s → 31,83min.**	
Syntaktische Richtigkeit	**4**	**1**	**2**	**3**	**4**	**14**
Semantische Richtigkeit	**4**	**1**	**2**	**4**	**3**	**14**
SUMME (15 möglich)	**11**	**3**	**6**	**9**	**9**	**38**

Anmerkungen

CodeFuse-DeepSeek-33B:

Actor-Relationship-Game:

Syntax: Bis auf einige, für die Richtigkeit des Codes syntaktisch irrelevante, fehlende Umbrüche für die Lesbarkeit gut.

In manchen Klassen fehlt das Information-Hiding und somit auch die damit verbundenen Getter und Setter.

Logische Richtigkeit: Gut, bis auf das in manchen Klassen fehlende Information-Hiding.

Lauffähigkeit: Leichte Klassen wie etwa die Actor und Movie Klassen können mit geringem Aufwand übernommen werden und sind lauffähig. Dort fehlen nur vereinzelt kleine Methoden. Komplexere Klassen können nur mit erhöhtem Mehraufwand lauffähig gemacht werden. Alles direkt zu kopieren führt zu keinem lauffähigen Stand.

Hone:

Syntax: Richtige Indentations, richtige Spaces, richtige Imports.

Logische Richtigkeit: Sieht sehr gut aus.

Lauffähigkeit: Läuft ohne Probleme und Anpassungen. Alle Tests (Unittests und Akzeptanztests) werden bestanden. Es wird dabei auch wirklich nicht die bereits existierende Python-Bibliothek „hone“ benutzt.

Hybrid-Images:

Syntax: Richtige Indentations, richtige Spaces, richtige Imports.

Logische Richtigkeit: Kaum etwas implementiert.

Lauffähigkeit: „demo.py“ funktioniert zwar. Es wurde jedoch sonst nichts implementiert.

Login-Registration:

Syntax: Sieht gut aus. Auch bei dem letzten Element wird ein Komma gesetzt. Es werden eher forEach als For-Schleifen genutzt. Es werden klassengebundene Aufrufe wie etwa JSON durchgeführt. Kaum Whitespaces. Auch die Imports sind richtig.

Logische Richtigkeit: Vue Komponenten erscheinen richtig. JavaScript Code weniger richtig.

Lauffähigkeit: Wenn alles kopiert wird nicht lauffähig. Allerdings funktioniert das Starten der Applikation, wenn die „webpack.config.js“ korrigiert wird. Dann fehlen jedoch die nötigen Input-Felder zu der Anmeldung. Es kann mit wenig Aufwand die App so erweitert werden, dass sie den Anforderungen entspricht.

Readtime:

Syntax: Richtige Indentations, richtige Spaces, richtige Imports.

Logische Richtigkeit: Sieht gut aus.

Lauffähigkeit: Alles funktioniert perfekt. Sogar richtige Kommentare für die Dokumentation. Es musste nur die „result.py“ Datei angepasst werden indem eine For-Schleife gelöscht wurde.

CodeQwen1.5-7B-Chat:

Actor-Relationship-Game:

Syntax: Es wurde nur die TMDB-API Klasse implementiert. Diese ist aber syntaktisch richtig.

Logische Richtigkeit: Nicht zu beurteilen, da kaum etwas generiert wurde.

Lauffähigkeit: Es wurde nur die TMDB-API Klasse generiert. Wird die generierte Implementierung der TMDB-API Klasse in das Referenzprojekt eingesetzt ist diese nur mit höherem Aufwand lauffähig.

Hone:

Syntax: Kaum etwas implementiert.

Logische Richtigkeit: Nicht zu beurteilen, da kaum etwas implementiert wurde.

Lauffähigkeit: Nicht gegeben es wurden nur die Methodenköpfe mit darauffolgenden „pass“ Befehlen implementiert. Dementsprechend schlägt auch schon das Ausführen der „demo.py“ fehl.

Hybrid-Images:

Syntax: Richtige Imports aber ansonsten kaum etwas implementiert.

Logische Richtigkeit: Nicht zu beurteilen, da kaum etwas implementiert wurde.

Lauffähigkeit: Es wurde nur die „demo.py“ implementiert und die funktioniert selbst nach Anpassungen nicht.

Login-Registration:

Syntax: Kaum etwas generiert. Dass was generiert wurde, erscheint richtig. Es wird auch bei dem letzten Element ein Komma genutzt. Es werden eher forEach als For-Schleifen genutzt und es existieren primär klassengebundene Aufrufe wie etwa JSON. Ebenso kaum Whitespaces. An Lineendings wird immer ein „;“ hinzugefügt.

Logische Richtigkeit: Nicht zu beurteilen, da kaum etwas generiert wurde.

Lauffähigkeit: Es wurde kaum etwas generiert. Somit auch nicht lauffähig.

Readtime:

Syntax: Richtige Indentations, Spaces und richtige Imports.

Logische Richtigkeit: Erscheint richtig.

Lauffähigkeit: Es laufen zwar alle Unittests und der Akzeptanztests. Jedoch scheitert das Ausführen der „demo.py“ Datei aufgrund der parse_html-Funktion in der „utils.py“ Datei. Es handelt sich dabei um einen TypeError, welcher nur mit einem erhöhten Aufwand behoben werden kann. Zudem liegt keine ausführliche Codedokumentation vor.

Nxcode-CQ7B-orpo:

Actor-Relationship-Game:

Syntax: Dass was generiert wurde, erscheint richtig. Jedoch wurde nicht so viel implementiert wie gefordert. Zu den Tests existieren auch nur die Methodenköpfe.

Logische Richtigkeit: Teilweise „return null“ bei wichtigen Methoden und viele Tests ohne richtige Implementierung, sondern nur mit Methodenköpfen.

Lauffähigkeit: Einfache Klassen mit kleinen Anpassungen an Typen und dem Hinzufügen von Setter-Methoden lauffähig. Es mussten auch Methoden umbenannt werden. Die Namen entsprachen nicht dem Oracle Standard. Komplexere Klassen benötigen einen sehr viel größeren Aufwand um lauffähig gemacht zu werden.

Hone:

Syntax: Kaum etwas implementiert

Logische Richtigkeit: Kaum etwas implementiert

Lauffähigkeit: Nichts implementiert und dementsprechend nicht lauffähig.

Hybrid-Images: Kaum etwas implementiert

Syntax: Kaum etwas implementiert

Logische Richtigkeit: Kaum etwas implementiert

Lauffähigkeit: „demo.py“ wurde implementiert. Funktioniert mit Anpassung der Imports (cv2 Import musste hinzugefügt werden). „hybrid.py“ wurde nicht implementiert.

Login-Registration:

Syntax: Kaum etwas generiert. Dass was generiert wurde, erscheint richtig.

Logische Richtigkeit: Kaum etwas generiert und das was generiert erscheint nur teilweise richtig.

Lauffähigkeit: Kaum etwas generiert. Nicht lauffähig.

Readtime:

Syntax: Teilweise unrichtige Indentations, welche zwar nicht zu Syntaxfehlern führen, aber ebenso nicht dem PEP8 Standard entsprechen.

Logische Richtigkeit: Will die bereits existierende „readtime“ Bibliothek verwenden, obwohl diese explizit bei den eingereichten Dokumenten ausgeschlossen wird.

Lauffähigkeit: Auch in diesem Fall laufen die Unittests und der Akzeptanztests. Allerdings läuft die „demo.py“ Datei nicht, da in den Methoden der „api.py“ Datei kein Default-Wert (ursprünglich 265WPM) hinterlegt wurde. In der „demo.py“ wird jedoch davon ausgegangen, dass dies der Fall sei wodurch der auftretende Fehler entsteht. Somit ist nach einer einfachen Anpassung der Code voll lauffähig. Auffällig ist, dass erneut keine ausführliche Dokumentation entstanden ist.

OpenCodeInterpreterDS-33B:

Actor-Relationship-Game:

Syntax: Standards werden eingehalten.

Logische Richtigkeit: Bei Tests sporadisch ohne Implementierung. Dass was da ist, erscheint aber richtig.

Lauffähigkeit: Bei einfachen Klassen mit geringen Anpassungen (Umbenennung von Methoden und Variablen, Schreiben neuer Setter) gegeben. Datentypen sind richtig. Bei komplexeren Klassen fehlen leider einige Methoden. In der GameplayInterface Klasse sind teilweise nur die Methodenköpfe implementiert und in den Methoden selbst nur Kommentare was dort zu implementieren wäre. Also ist insgesamt nur bedingt eine Lauffähigkeit gegeben.

Hone:

Syntax: Richtige Imports richtige Indentations und Spaces.

Logische Richtigkeit: Erscheint gut. Es fehlt jedoch Code und zudem wird teilweise versucht die „hone“ Bibliothek zu nutzen, obwohl diese in den Eingabedokumenten explizit ausgeschlossen wurde.

Lauffähigkeit: Funktioniert nicht direkt. Es müssen die Klassenvariablen für „hone.py“ angepasst werden, um keinen Fehler zu erhalten. Dann erscheint ein Fehler mit durch einen Rückgabewert „null“. Somit ist selbst mit leichten Anpassungen das Programm nicht lauffähig. Es müsste somit mehr Zeit investiert werden.

Hybrid-Images:

Syntax: Kaum Code.

Logische Richtigkeit: Kaum Code.

Lauffähigkeit: „demo.py“ wurde implementiert und funktioniert auch ohne Anpassungen. „hybrid.py“ wurde nicht implementiert.

Login-Registration:

Syntax: Nicht so gut wie die vorherigen Modelle. An den letzten Elementen fehlt das Komma. Manchmal keine richtige Verwendung von „const“. Auch fehlende „;“und teilweise keine dedizierten Imports.

Logische Richtigkeit: Erscheint gut. Es wurde viel Code generiert.

Lauffähigkeit: Es wurde viel Code generiert. Dieser ist aber nicht lauffähig. Auch nach weiteren Anpassungen konnte das Programm nicht gestartet werden.

Readtime:

Syntax: Erscheint richtig. Richtige Imports und richtige indentations. Jedoch wird erneut versucht die bereits vorhandenen Implementierungen der Bibliothek „readtime“ zu nutzen, obwohl diese explizit nicht als nutzbare Bibliothek erlaubt war.

Logische Richtigkeit: Erscheint richtig.

Lauffähigkeit: Es ist ohne eine einzige Anpassung lauffähig. Sowohl alle Tests (Unit und Akzeptanztests) als auch die „demo.py“ Datei.

Phind-Codellama-34b-v2.0:

Actor-Relationship-Game:

Syntax: Viele single line Kommentare passen nicht gemäß dem von den Oracle Standards vorgegebenen Schema „/* */“.

Logische Richtigkeit: Einfache Klassen werden richtig implementiert. Sobald komplexere Klassen implementiert werden sollen kommt das Modell nicht mehr damit klar und setzt für die eigentliche Implementierung nur Platzhalterkommentare.

Lauffähigkeit: Es fehlen viele Methodenimplementierungen. Häufig nur Methodenköpfe mit Kommentaren. Die einfachen Klassen können erneut unter Anpassungen der Methoden und Attributnamen sowie einiger Datentypen wieder lauffähig gemacht werden.

Hone:

Syntax: Erscheint richtig.

Logische Richtigkeit: Sieht ebenso gut aus aber es fehlt etwas mehr Code vor allem bei der „hone.py“ Datei. Es wird nicht versucht die existierende Python Bibliothek „hone“ zu nutzen. Es werden sogar JSON Dateien generiert, welche als Beispiel dienen sollen.

Lauffähigkeit: Ist nicht lauffähig. Auch nach einigen Anpassungen an der „hone.py“ nicht lauffähig.

Hybrid-Images:

Syntax: Falsche Imports. Es werden keine dedizierten Imports genutzt, sondern mit Wildcard Imports die Python Bibliothek „hybrid“ importiert. Dies sollte basierend auf den Eingabedaten nicht erlaubt sein.

Logische Richtigkeit: Erscheint richtig.

Lauffähigkeit: „hybrid.py“ wurde nur mit Funktionsköpfen implementiert. Somit sind innerhalb dieser Methoden nur Kommentare zu finden. „demo.py“ wurde implementiert und funktioniert ohne Anpassungen.

Login-Registration:

Syntax: Erscheint richtig. Richtige Kommata gesetzt und richtige dedizierte Imports.

Logische Richtigkeit: Erscheint gut. Kleine Details wie zum Beispiel die Angabe der minimalen Länge des Passworts mit sechs ist nicht vorhanden.

Lauffähigkeit: Da die App mit nicht vorgesehenen Bibliotheken implementiert wurde, mussten diese installiert werden. Nach der Installation von beispielsweise „axios“ startet die App zwar laut Logs ohne Probleme. Anscheinend ist aber die Referenz zu den Vue-Komponenten nicht richtig, da lediglich ein Whitescreen im Browser erscheint. Auch nach weiteren Anpassungen war es nicht möglich die Vue Komponenten richtig zu importieren.

Readtime:

Syntax: Entspricht den Standards. Es werden keine Wildcard Imports genutzt.

Logische Richtigkeit: Es wird ebenso aus der existenten Bibliothek „readtime“ importiert, obwohl dies nicht getan werden soll. Ansonsten erscheint es logisch richtig.

Lauffähigkeit: Es funktioniert fast alles. Die „demo.py“ wirft jedoch an dem Ende einen Fehler, dass das Result-Objekt kein Attribut „seconds“ besäße. Selbst nach dem Entfernen dieses falschen Verweises und der Anpassung der „demo.py“ konnte jedoch keine Lösung gefunden werden. Es fehlen zudem ein paar Kommentare für die Codedokumentation.