Aufgabenstellung im WS 2019/2020

Cloud Computing Technology

Prof. Dr.-Ing. Christoph Kunz & Prof. Dr.-Ing. Peter Thies

Modul #:	Prüfungsleistung:	Workload:
368101	Praktische Arbeit und Präsentation (PP)	5 ECTS

1. Einleitung

Ziel dieser Lehrveranstaltung ist eine vertiefende Auseinandersetzung mit technischen Fragestellungen des Cloud Computing und affinen Technologien. Einerseits sollen sich die Teilnehmer spezifische Techniken aneignen und damit auseinandersetzen. Andererseits sollen diese Techniken von allen Studierenden im Kontext eines Projekts angewendet werden. Da Software stets das Ergebnis von Teamaktivitäten ist, sollen die Projektarbeiten im Gruppenkontext durchgeführt werden.

Im Rahmen der Projektarbeit steht insbesondere die Frage im Vordergrund, wie am Markt verfügbare Softwarekomponenten unter technischen und wirtschaftlichen Gesichtspunkten zu einem Gesamtsystem assembliert werden können. Dabei ist ein Customizing jener Komponenten bzw. eine entsprechende individuelle Implementationstätigkeit unumgänglich. Weiterhin ist eine wirtschaftliche Beurteilung der verwendeten Dienste bzw. Komponenten erforderlich.

2. Szenario

Lisa Meier ist Mitarbeiterin im Projekt HighNet, zur Erarbeitung neuer Ansätze der Integration von Informationslogistik für z.B. PKW-Fahrer. Im Rahmen dieses mehrjährigen Vorhabens arbeitet Ihre Stuttgarter Firma Broken Bits für das weltweit agierende Unternehmen Star Cars in Tokio. Das Projekt ist eines von vielen, dass Broken Bits sowohl mit Star Cars als auch anderen Auftraggebern der Branche verbindet.

Lisa Meier ist eine von 3 Probanden zur Erprobung von *MyCorrespondent* (kurz: MyCo). Bei dieser von Broken Bits konzipierten und realisierten Cloud-basierten Dienstleistung handelt es sich um einen intelligenten Sprachassistenten. *MyCo* verbindet die Möglichkeiten von Assistenten wie Google Assistant bzw. Google Home mit Verfahren zur Realisierung künstlicher Intelligenz.

Mittels *MyCo* kann Lisa nun im angebundenen Nachrichten-Repository recherchieren. Dabei kann *MyCo* anders als heutige Systeme wie Amazon Alexa oder Google Home einen langfristigen Gesprächskontext bzw. ein persönliches Interessenprofil erlernen und zur Anwendung bringen, während es sich mit Lisa unterhält oder Antworten gibt.

Im Zuge des Projekts HighNet werden zahlreiche Dokumente bereitgestellt. Insbesondere Dokumente von Dritten (vgl. Nachrichten) spielen eine Rolle. Dokumente werden vorrangig in einer Markup-Sprache repräsentiert. Es können jedoch auch Dokumente der folgenden Typen zur Anwendung kommen: Office-artige Textdokumente, Tabellen sowie Grafiken, Videos, Software-Quelltext, Links auf Web Pages u.v.a.m.

Heute ist wieder einer der Tage, an dem Lisa eine Testkonversation mit dem *MyCo*-Prototypen durchführt. Sie initiiert dazu eine Konversation mit *MyCo*. Dieses System setzt auf einem verbreiteten Sprachassistenzsystem auf und bietet einen neuen, viel weitergehenden Service.

Während Sie sich mit *MyCo* über das Projekt unterhält, interpretiert die Software das Gesprochene. Dies resultiert in einer separaten Anzeige von Projektinhalten, die wahrscheinlich mit dem gerade Besprochenen in Verbindung stehen. Durch Sprachbefehle kann sie das jeweilige Dokument anzeigen lassen oder annotieren (z.B. durch neue Schlagwörter oder allgemeine Anmerkungen):

Prof. Dr.-Ing. Peter Thies
Hochschule der Medien
Nobelstraße 10
70569 Stuttgart
✎ thies@hdm-stuttgart.de
☎ +49 (0) 711 / 89 23 - 31 94
⌂ www.prof-thies.de

Lisa: *Hello MyCo!*
MyCo: *Hello, Lisa! How are you?*
Lisa: *I'm fine. Thank you.*
[Gesprächshistorie von Lisas Gesprächen mit MyCo][1]
Lisa: *Let´s talk about current political activities concerning health care within the USA.*
MyCo: *Ok, Lisa, what do you want to know?*
Lisa: *Show me all the documents of the Obama presidency!*
MyCo: *No problem. Here they are!*
[Nachrichtendokumente]
Lisa: *Please show me the docs that are still relevant under the actual government!*
MyCo: *Wait a second - do you expect these?*
[Nachrichtendokumente]
Lisa: *Yeah, but there's one document missing. I need the doc we've been talking about yesterday. It was a comparison of the US' health system and those of other industrial nations.*
[-]
MyCo: *Yesterday we used those two reports. Which one do want to see?*
[Zeigt beide Reports an und nennt zudem deren Namen akustisch.]
Lisa: *Good boy! Show me "Document 879". That's what I've been looking for!*
MyCo: *Here you are!*
[Das entsprechende Dokument wird angezeigt. Lisa liest und ist stumm.]
MyCo: *Lisa, what else can I do?*
Lisa *Oh, thank you! That's all for now. Bye!*
MyCo: *You are welcome. Have a nice day! Bye!*
[MyCo beendet die Konversation.]

Das Gespräch verläuft also erfolgreich. Lisa findet, was sie sucht.

Trotz anfänglicher Vorbehalte konnte man sich auf den Einsatz des neuen Assistenzsystems einigen, das vollständig auf Google-basierter Software aufsetzt. So kommen etwa Google Home/Assistant, Tensorflow, Google Drive mit den darin bekannten Dokumenttypen sowie Google Container Engine, Google Compute Engine, Google App Engine sowie die Speicherlösungen von Google, wie etwa Google Cloud Storage zum Einsatz.

3. Rahmenbedingungen

Die Teilnehmer sollen in einer simulierten Umgebung einen technologischen Innovationsprozess erleben und gestalten. Es wird angenommen, dass es sich bei der Umgebung um ein Unternehmen handelt, dass im Rahmen einer neuen Geschäftsidee den Einstieg in neue Technologien (Sprachassistenten in Verbindung mit KI[2]) wagt. Auch wenn die finale Programmierung des Zielsystems wahrscheinlich bei einem Zulieferer/Auftragnehmer (Inland oder Offshore) stattfinden wird, so muss doch der Aufbau von geschäftskritischem Know How (vgl. Intellectual Property) vor Ort erfolgen, um Spezifikationen für Auftragnehmer erstellen und diese effektiv steuern zu können.

Es wurde ein kleines Projektteam gegründet, das auf Basis von vorgefertigter Software (z.B. Google Infrastruktur) mit möglichst hoher Produktivität Machbarkeitsstudien im Bereich zuvor identifizierter „Challenges" durchführt. Das Vorgehen sowie die Ergebnisse dieser Studien müssen für nachfolgende Projektphasen und später hinzuzuziehende Projektmitarbeiter (in-/extern) hinsichtlich Inhalt und Struktur geeignet aufbereitet und dokumentiert werden. Darauf aufbauend wären zu einem späteren Zeitpunkt Spezifikationen erforderlich, die an Zulieferer im Rahmen einer Beauftragung weitergegeben werden könnten.

[1] Texte in eckigen Klammern beschreiben mögliche Dokumente, die automatische zur Auswahl angezeigt werden.
[2] KI = Künstliche Intelligenz

4. Aufgaben

Die Zusammensetzung der Gruppen wird wird den Studierenden überlassen. Alle Teilnehmer müssen spätestens eine Woche nach Bekanntgabe der Aufgabe in den begleitenden Moodle-Kurs eingetragen sein und die Zusammensetzung ihrer Gruppe gemeldet haben, um die Prüfungsleistung in diesem Semester erbringen zu können.

Im Projektverlauf muss jede Gruppe Teilaufgaben zusammen mit den dafür verantwortlichen Gruppenmitgliedern definieren. Zum Zeitpunkt des zweiten Meilensteins (s. Tabelle am Ende dieses Dokuments) müssen die für die einzelnen Teilaufgaben verantwortlichen Gruppenmitglieder benannt werden.

Die Beurteilung der praktischen Arbeit (vgl. Prüfungsform PP) erfolgt durch die Dozenten. Jedes Teammitglied erhält eine individuelle Bewertung. Dennoch steht die gesamte Gruppe wie im beruflichen Alltag einem kollektiven Erfolgszwang gegenüber.

Zum erfolgreichen Bestehen der praktischen Arbeit muss jeder der in der Tabelle am Ende dieser Ausführungen dargestellten Meilensteine erbracht werden. Die Note der Prüfungsleistung wird anhand der individuellen Teilleistungen gemäß Tabelle 1 ermittelt.

Die für die erfolgreiche Bearbeitung des Projekts nötigen Kenntnisse bauen auf den Zugangsvoraussetzungen zum Studiengang auf und müssen selbständig erweitert werden.

Zur Unterstützung der Projektteams werden Projektmeetings mit den Dozenten abgehalten, die zur vereinbarten Zeit in Anspruch genommen werden können. Ansonsten erfolgt die Teamarbeit (Recherche, Dokumentenerstellung und Programmentwicklung) in eigener Regie.

5. Charakterisierung des Zielsystems

Jede Gruppe hat die Aufgabe, im Rahmen von Fallstudien Komponenten eines Cloud-basierten Sprachassistenzsystems zu entwickeln, das dem in Abschnitt 2 vorgestellten Szenario entspricht bzw. hierfür eine angemessene Lösung bietet. Das Szenario bietet einige Herausforderungen, die zu bewältigen sind. Hierzu sind Recherchen, Machbarkeitsstudien und schließlich Bewertungen erforderlich, wie man als Team diese Challenges meistern will. Folgende Herausforderungen müssen von den Teilnehmern bewältigt werden:

Challenge 1: Nutzung herkömmlicher Sprachassistenten

Aus der Konversation mit dem Assistenten müssen Gesprächsinhalte abgeleitet werden, die eine Identifikation von Gesprächsthemen ermöglichen.

Challenge 2: Repräsentation des Gesprächskontexts sowie der Dokumentstrukturen

Die in den Nachrichtendokumenten dargestellte Information muss hinsichtlich ihrer Bedeutung für den Gesprächskontext aufbereitet werden. Die Frage, welche Funktion ein Dokument bezüglich eines Gesprächskontexts besitzt, muss geeignet repräsentiert werden, um mittelbar die thematische Zuordnung des jeweiligen Dokuments zu einem Gesprächskontext zu ermöglichen.

Die Verbindung zwischen Spracherkennung, Dokumentennutzung und Gesprächskontext erfordert die Repräsentation von Gesprächskontexten z.B. auf Basis von Ereignissen.

Challenge 3: Maschinelles Lernen

Mit der Zeit sollte sich das System bei der Präsentation relevanter Dokumente bezogen auf den Gesprächskontext verbessern. Vom Nutzer in einem Gesprächskontext erwähnte, verwendete, d.h. ausgewählte oder geöffnete Dokumente zeichnen diese besonders aus. Diese Zusammenhänge können mit der Zeit gelernt und für die Auswahl von zu präsentierenden Dokumente genutzt werden.

6. Spezifische Anforderungen an das Zielsystem

An das zu realisierende System werden die folgenden Anforderungen gestellt:

A1: Das System soll das Auswählen und Anzeigen von Dokumenten aus einem Repository (z.B. zahlreiche XML-Dokumenten oder Google Drive/Google Docs, wie Textdokumente, Tabellen, Präsentationen) während Konversationen unterstützen.

A2: Eine Konversation wird mittels eines Sprachassistenten auf Basis von Google Home/Assistant durchgeführt.

A3: Während einer Konversation wird das Gespräch analysiert. Es wird daraus der Gesprächskontext (z.B.: Wer spricht aktuell worüber? Wann hat man worüber gesprochen und welche Dokumente wurden konsumiert?) abgeleitet und einer weiteren Verarbeitung (Lernen) zugänglich gemacht.

A4: Die im Dokumenten-Repository abgelegten Dokumente werden inhaltlich hinsichtlich ihres Typs (z.B. Nachricht aus der Wirtschaft, Unfallmeldung usw.) analysiert. Es wird eine entsprechende inhaltliche Repräsentation für diese Dokumente softwaretechnisch erstellt und abgespeichert.

A5: Eine Inferenz-Komponente leitet aus den bzgl. A3 & A4 erzeugten Daten während einer Konversation eventuell relevante Dokumente ab und benachrichtigt den Benutzer.

A6: Eine zur Konversation ergänzende, parallel ausgeführte Zusatzkomponente zeigt bzgl. A5 alle relevant eingestuften Dokumente zur Auswahl an (einfache Liste wie in einem File Manager).

A7: Einzusetzende Systeme bzw. Dienste sind zwingend Google Home/Assistant, Google Drive sowie ergänzend ggf. Google Compute Engine, Google Container Engine, Google App Engine und Google Web Toolkit. Der Einsatz weiterer Systeme oder Dienste ist im Vorfeld mit den Dozenten abzustimmen.

A8: Für die Dokumentation von Software-Strukturen ist die Unified Modeling Language (UML) zu verwenden. Dies gilt insbesondere für den Software-Entwurf.

A9: Für die Inferenz im Allgemeinen (siehe A5) ist Tensorflow zu verwenden.

A10: Verwendete Programmiersprachen: Sofern eine Java API oder Python API der verwendeten Dienste und Komponenten verfügbar ist, sind diese zu nutzen. Ergänzend kann auch auf Javascript zurückgegriffen werden. Hierzu muss innerhalb der Gruppe ein Konsens erzielt werden.

A11: Falls erforderlich sind für die Integration (Nutzung und Bereitstellung) von Diensten Web Services nach dem REST-Ansatz zu nutzen. Verfügbare Frameworks, die dies z.B. im Java-Umfeld erlauben, sind Jersey, RestEasy, Restlet und Google Cloud Endpoints.

A12: Jegliche Dokumente sind in dem Format Markdown zu verfassen.

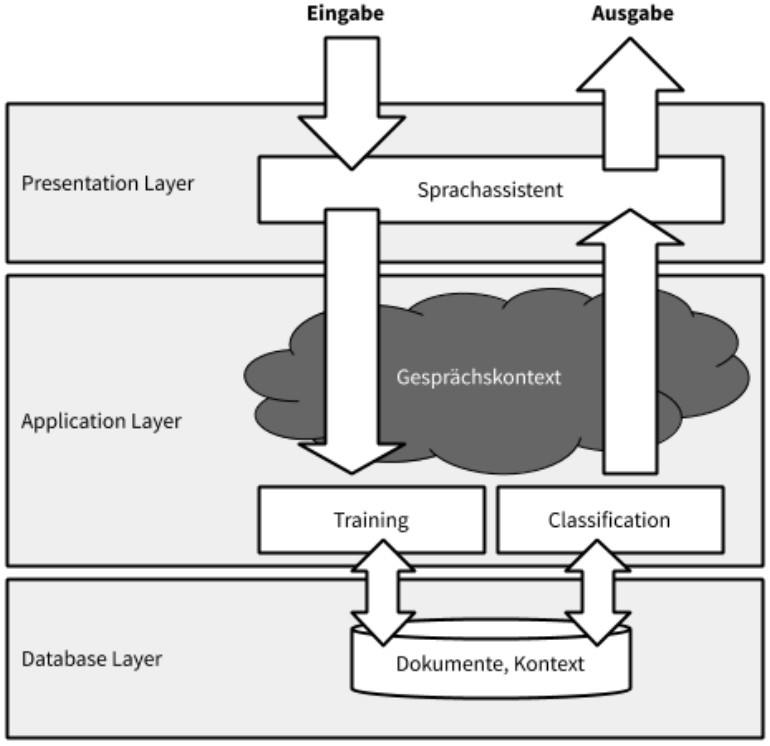

Abbildung 1: Grobarchitektur des Zielsystems

Es ist ein mehrschichtiges System[3] zu entwickeln, das aus einer Vielzahl von Komponenten und Diensten bestehen wird (vgl. Abbildung).

Die oberste Schicht wird die Präsentationsschicht sein. Sie wird dem Sprachassistenten und dessen Anbindung sowie dem Watchdog bestehen. Mit Hilfe des Watchdog werden Ereignisse einer Konversation zu einem Gesprächskontext zusammengefügt und so der Übergang zur Applikationslogik geschaffen.

Die mittlere Schicht nimmt die Applikationslogik (auch Anwendungslogik oder Business-Logik genannt) auf. Sie enthält sämtliche Dienste, die notwendig sind, um die Datenstrukturen (Dokumente, Ereignisse, etc.) und Funktionen (Auswahl relevanter Dokumente, Feststellen von Gesprächsgegenständen etc.) des Systems handhaben und bereitstellen zu können. Damit erfolgt eine Trennung von den anderen Funktionen des Systems, die sich z.B. mit der äußeren Darstellung oder der internen Speicherung der Daten befassen. Für eine Applikationslogik ließen sich somit mehrere Präsentationslogiken und mehrere Speicherlogiken definieren.

Anders als in monolithischen Client-Server-Architekturen muss hier die mittlere Schicht nicht durch einen einzelnen Applikationsserver dargestellt werden, sondern kann aus eine Vielzahl von Diensten (Web Services) bestehen, die unabhängig voneinander lauffähig sein sollen. Es sind mindestens die in der Abbildung (s.o.) abgebildeten Dienste anzubieten.

Die oben genannte Funktionalität erfordert, dass sich die Applikationslogik in Kontakt mit dem Repository (z.B. Google Drive) befindet, um die jeweils nötigen Daten/Dokumente auslesen bzw. neue Daten (s. Gesprächskontext) ablegen zu können. Die Datenbankschicht wird demzufolge mit einem Repository sowie dessen Anbindung an die Services der Applikationsschicht realisiert. Die Anbindung des Repository kann durch Google-spezifische Frameworks in Verbindung mit eigenentwickelten Mappern erfolgen.

Die Gruppen sollen dafür sorgen, dass möglichst viele Teile des Gesamtsystems in der Cloud laufen. Der Client ist auf einem beliebigen Rechner, der auf die Cloud zugreifen kann, vorzuführen. Die Teilnehmer

[3] Man spricht in diesem Zusammenhang auch von einer 3-Tier- oder Multi-Tier-Architektur.
Tier wird ['tir] ausgesprochen und steht im Englischen für: Reihe, Sitzreihe, Rang.

haben dafür Sorge zu tragen, dass die Demonstration vorab in dem ihnen genannten Raum lauffähig ist. Hierzu bietet sich ein Test an. Der Raum wird den Teilnehmern rechtzeitig vorher bekannt gegeben. Termine für Tests sind rechtzeitig zu vereinbaren.

Die Aufteilung in drei Schichten bietet einige wichtige Vorteile gegenüber z.B. zweischichtigen Ansätzen. Zum einen wird durch die Konzentration sämtlicher Applikationslogik auf die mittlere Schicht eine Entkopplung der Präsentationsschicht von der Datenbank erreicht. Zum anderen können Präsentations-Clients verändert werden[4], ohne die Applikationslogik modifizieren zu müssen. Darüber hinaus ist es so möglich, auf derselben Applikationslogik unterschiedliche Clients aufsetzen zu können. Dies könnten einerseits Clients bzgl. unterschiedlicher Medien (Sprachausgabe, Web-basiert oder dedizierte Applikationen) oder bzgl. unterschiedlicher Benutzerrollen (z.B. für Reporting oder Systemnutzung) sein. Aus diesem Grund ist unbedingt darauf zu achten, dass Clients keinerlei Applikationslogik enthalten. Diese ist ausschließlich in der Applikationsschicht zu finden!

7. Vorgehensweise

Im Rahmen eines professionellen Software-Engineerings wird die Einhaltung eines Software-Entwicklungsprozesses erwartet. Dies bedeutet, dass die Projektarbeit in festgelegten Schritten abläuft (z.B. Anforderungsanalyse, Systemspezifikation/Entwurf, Implementierung, Test). Hierbei kann ein wasserfallartiges Vorgehen oder ein agiler Ansatz (z.B. Extreme Programming oder Scrum) gewählt werden.

Während jeder Projektphase ist auf eine angemessene Dokumentation zu achten. Hierzu gehören z.B. die Erstellung von UML-Diagrammen sowie deren Erläuterung und die ausführliche, sinnvolle Dokumentation des Programmcodes. Im Falle von Java ist Javadoc zu verwenden.

Es ist wichtig, dass man den Entwicklungsprozess während und auch nach Abschluss des Projekts nachvollziehen kann. Daher ist auch eine textuelle Beschreibung inklusive Installations- und Bedienungsanleitung nötig.

Für die Erstellung von UML-Diagrammen ist StarUML zu verwenden. Zur Erstellung von Source-Code ist im Falle von Java die Entwicklungsumgebung Eclipse oder im Falle von Python PyCharm zu verwenden. Es wird erwartet, dass der Source-Code den Konventionen der jeweils verwendeten Sprache entsprechend erstellt wird. Im Fall von Java wären dies z.B. die Java Code Conventions, im Fall von Python PEP 8. Sämtliche verwendeten Frameworks bzw. APIs müssen geeignet dokumentiert und hinsichtlich ihrer Verfügbarkeit (lizenzrechtlich und wirtschaftlich) analysiert und dies dokumentiert werden.

Zur Koordination der Projektteilnehmer untereinander und den verschiedenen Software-Entwicklungsstufen ist *Git* als Werkzeug zur verteilten Versionsverwaltung unter Nutzung des webbasierten Git-Hosting-Dienstes GitHub zu verwenden. Für Eclipse existiert dazu das eGit-Plugin, mit dessen Hilfe die Git-Funktionen innerhalb der Entwicklungsumgebung genutzt werden können. PyCharm besitzt eine analoge Funktionalität. Die GitHub-Nutzernamen müssen unbedingt eine Identifizierung der Teilnehmer *in Form von Klarnamen* ermöglichen. Abkürzungen und Pseudonyme werden nicht akzeptiert. Den Dozenten[5] ist spätestens ab Meilenstein 2 dauerhaft ein Zugriff auf sämtliche GitHub-Repositories einzuräumen, die im Zuge dieses Projekts benutzt werden. *Jedes Teammitglied hat sich an der Entwicklungstätigkeit substantiell und nachvollziehbar zu beteiligen. Die individuellen Beiträge sind in mindestens einem Commit pro Arbeitstag und Teammitglied in GitHub zu fixieren. Auch sämtliche weiteren Dokumente sind in Github abzulegen. Diese müssen wie oben bereits erwähnt in Markdown-Format vorliegen. Etwaige bildliche Darstellungen sind als separate Bilddateien in Github abzulegen und geeignet in die Markdown-Dokumente einzubetten. Es werden nur solche individuellen Beiträge gewertet, deren zugehörige Commits eindeutig und offensichtlich einem Git-Benutzerkonto unter den sogenannten "Contributions" in Github zuordenbar sind.*

Im Rahmen von vorausgegangenen Studien der Teilnehmer wie z.B. Bachelor-Studium der Wirtschaftsinformatik wurden die für die erfolgreiche Projektbearbeitung erforderlichen Grundlagen vermittelt. Es wird erwartet und ist integraler Bestandteil der Lehrveranstaltung, dass selbständig

[4] Z.B. wenn eine Client-Software bzgl. ihres Erscheinungsbilds (optisch oder akustisch) überarbeitet werden soll
[5] GitHub-Id: PeterThies & ChristophKunz

weiterführende Informationen aus der Literatur und im WWW genutzt werden.

8. Bewertungskriterien

Die Bewertung des Arbeitsergebnisses erfolgt aufgrund unterschiedlicher Aktivitäten: Zum einen werden nach Abschluss des Meilensteins 3 und des Meilensteins 6 (vgl. Tabelle 1) Präsentationen bzw. Demonstrationen der bis dahin erreichten Arbeitsergebnisse durchgeführt. Bei der Bewertung werden die Leistungen der Studierenden gemäß dieser Aufgabenstellung bewertet. Hierzu gehört die Beteiligung an der Erstellung der Projektdokumente und insbesondere die Source-Code-Beiträge in GitHub.

Wichtig: Die Teilnehmer haben die Ergebnisse selbständig zu erbringen und sich an sämtlichen Tätigkeiten umfänglich und substanziell zu beteiligen. Zu jeglichen Komponenten (sämtliche Programmteile und Dokumentationen) ist deren Urheberschaft zu dokumentieren. Hierzu gehört auch, dass sämtliche Quellen in geeigneter Weise genannt und adäquat gekennzeichnet werden.

Maßgeblich für die Deadlines zur Erreichung der Meilensteine sind die in der Tabelle 1 dargestellten Termine. Eine spätere Abgabe und eine zu Github *alternative Abgabe ist ausgeschlossen*.

9. Weitere Informationen

Weitere Informationen finden sich im Moodle-Kurs für diese Lehrveranstaltung. Insbesondere sind die FAQ des Moodle-Kurses Teil dieser Aufgabenstellung.

Tabelle 1: Meilensteine & Bewertungskriterien

	Meilenstein	Bewertung	Bewertungskriterien	Form	Termin
1.	Teilnahme an der Auftaktveranstaltung Registrierung in Moodle	bestanden/ nicht bestanden	Hat die Person sich in den Moodle-Kurs eingetragen und an der Aufgabenbesprechung teilgenommen?	pers. Teilnahme, Login in den Moodle-Kurs	22.10.2019
2.	Aufstellung von Teilaufgaben und Verantwortlichkeiten; Einrichten von GitHub-Repositories und dauerhafter Zugriff für Dozent und Teammitglieder	bestanden/ nicht bestanden	Inhalt und Umfang der Beiträge, erkennbare Eigenleistung.	Dokumente auf Github	29.10.2019
3.	Realisierung Fallstudie Google Assistant, Dialogflow	40 Punkte	Anschaulichkeit, Aussagekraft, Nachvollziehbarkeit sowie Relevanz und Adäquanz der Fallstudie. Inhalt und Umfang der individuellen Beiträge, erkennbare Eigenleistung.	Source Code und weitere Dokumente auf Github	03.12.2019
4.	Demonstration / Präsentation	5 Punkte	Struktur, Anschaulichkeit, Angemessenheit und Transferqualität der Präsentation/Demonstration. Nachweis der allgemeinen Funktionalität der Fallstudie.	Demonstration / Präsentation; pers. Teilnahme; Dokumente auf Github	03.12.2019
5.	Peer-Feedback	bestanden/ nicht bestanden	Peer Feedback abgegeben; Tiefe der Auseinandersetzung, Strukturierungsgrad	Datei in Moodle	03.12.2019
6.	Realisierung Fallstudie Tensorflow	70 Punkte	Anschaulichkeit, Aussagekraft, Nachvollziehbarkeit sowie Relevanz und Adäquanz der Fallstudie. Inhalt und Umfang der individuellen Beiträge, erkennbare Eigenleistung.	Source Code und weitere Dokumente auf Github	30.01.2020
7.	Demonstration / Präsentation	5 Punkte	Struktur, Anschaulichkeit, Angemessenheit und Transferqualität der Präsentation/Demonstration Nachweis der allgemeinen Funktionalität der Fallstudie.	Demonstration / Präsentation pers. Teilnahme; Dokumente auf Github	30.01.2020
8.	Peer-Feedback	bestanden/ nicht bestanden	Peer Feedback abgegeben; Tiefe der Auseinandersetzung, Strukturierungsgrad	Datei in Moodle	30.01.2020