Transparent Skills Alignment Mechanism - CV, Interview Wizard & Heatmap

Projektdokumentation – 40

Sebastian Laux <sebastian.laux@stud.tu-darmstadt.de>
Philip Pfaender <philip.pfaender@stud.tu-darmstadt.de>
Alexander Zeikowski <alexander.zeikowski@stud.tu-darmstadt.de>
Dennis Weinberger <dennis.weinberger@stud.tu-darmstadt.de>
Maximilian Stillger <maximilian.stillger@stud.tu-darmstadt.de>

Teamleitung:
Irem Diril <haticeirem.diril@stud.tu-darmstadt.de>

Auftrag:
Dr. Michael Nofer <michael.nofer@direktberatung.de>
DIErektBERATUNG AG
info@direktberatung.de

31. März 2021

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bachelorpraktikum
Wintersemester 2020/21
Fachbereich Informatik

Inhaltsverzeichnis

1. Projektbeschreibung

1.1. Beschreibung

1.1.1. Motivation

Das *Transparent Skills Alignment Mechanism* Projekt ist eine Webapplikation zur Unterstützung des Berufsbewerbungsprozesses/Assetmanagements für Firmen, welche in der Bewertung und Wahl von Bewerber*innen assistieren soll.
Firmen erhalten je nach Unternehmensgröße teils eine hohe Anzahl an Bewerbungen, die nebenbei nur schwierig zu bewerkstelligen sind, ohne extra Personal einzustellen. Bewerbungen sind von den verschiedensten Menschen und weisen viele verschiedene Grafikdesigns und einige Formate auf, die für den*die Leser*in die schnelle Vergleichbarkeit erschweren. Eine Software kann hierbei eine Hilfe sein, um zum einen das Extrahieren wesentlicher Daten über bestimmte Fähigkeiten (Skills) an den*die Bewerber*innen auszulagern und zum anderen einen schnellen Überblick über die Skills möglicher Kandidaten*innen zu erhalten und entsprechend eine Vorauswahl treffen zu können.
Mit einem System, welches seine Bewertung auf Basis von wesentlichen Fähigkeiten erstellt, entsteht eine fairere Evaluation, als wenn Leute zum Beispiel auf Grund eines auffälligen Bewerbungsdesigns ins Auge stechen.
Für unseren Arbeitgeber soll unser Projekt die Grundstruktur für die Software bilden.

1.2. Umsetzung

1.2.1. Aufbau

Als Quellen der Bewertungen dienen unter anderem ein *Interview Wizard*, *CV Wizard* sowie eine Anbindung an Drittplattformen. Diese werden gewichtet und leicht interpretierbar in der *Decision Heatmap* visualisiert.

1.2.2. Komponenten

Interview Wizard

Der Interview Wizard soll dabei unterstützen die verschiedenen Bewerber*innen im persönlichen Gespräch einzuschätzen. Dafür werden die Interview-Fragen aufgelistet, zu denen sich der*die Interviewer*in direkt im Gespräch Notizen (sogenannte *Observations*) erstellen kann. Außerdem gibt es die Möglichkeit, dass weitere Observer*innen passiv am Interview teilnehmen und sich mittels eines Chats untereinander austauschen können. Schließlich bekommen die Bewerber*innen für ihre einzelnen Fähigkeiten eine Bewertung.

CV / Document Wizard

Mit dem CV Wizard können Bewerber*innen ihren Lebenslauf hochladen und , in Form eines Chats, eine Selbsteinschätzung ihrer Fähigkeiten abgeben. Es gibt die Möglichkeit, Ausschnitte im Lebenslauf zu markieren,

die eine Aussage über die Fähigkeiten des*der Anwärter*in tätigen. Dadurch können Bewerber*innen ihre Selbsteinschätzung direkt mit Textstellen belegen.

Decision Heatmap

Die Decision Heatmap (Abbildung 1.1) ist die zentrale Komponente des Projekts. In der obersten Zeile sind die Bewerber*innen gelistet und in der linken Spalte die Fähigkeiten (Skills). Hierbei können Skills eine beliebig unterschiedliche Gewichtung aufweisen. Entsprechend der Ergebnisse aus den Wizards werden die Felder der Matrix eingefärbt (grün = gut, rot = schlecht, mit Zwischenstufen) und ergeben eine visuelle Hilfestellung für den Jobersteller, um eine schnellere Vorauswahl treffen zu können. Dabei werden die Kandidat*innen gemäß ihrer berechneten Gesamtnote vorsortiert. Per Mausklick auf die Matrixelemente ist es zudem möglich, die Daten anzeigen zu lassen, die zu der entsprechenden Bewertung führen.

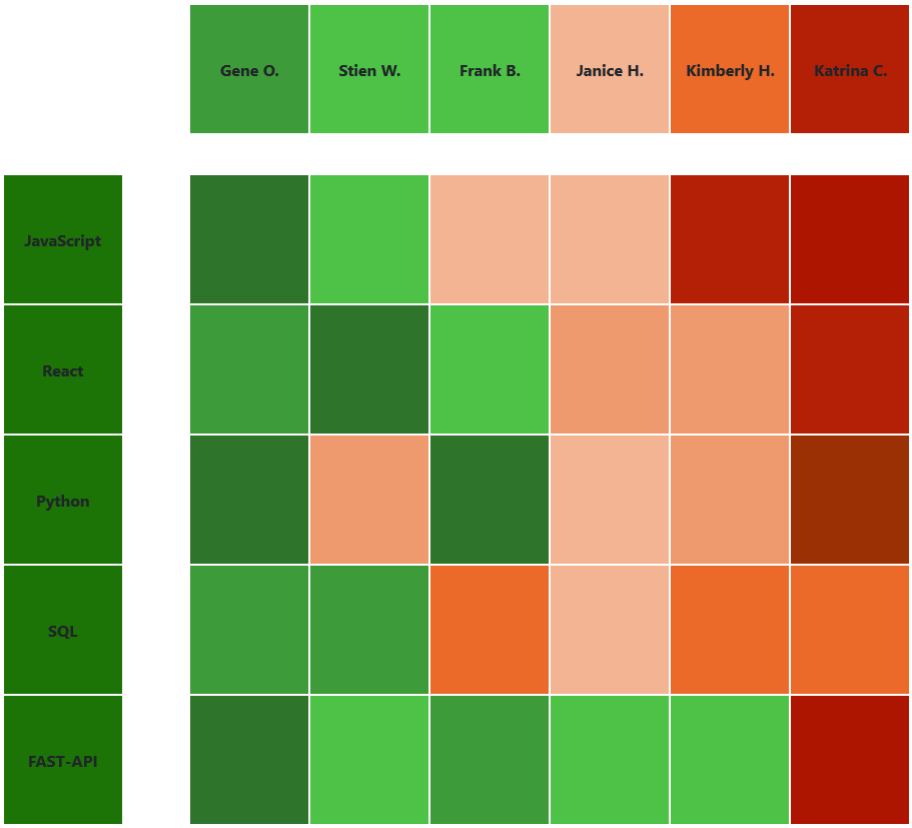

Abbildung 1.1.: Decision Heatmap

1.2.3. Tech Stack

Frontend

Im Frontend wird die JavaScript Library *React.js* verwendet, welche gut geeignet ist, um interaktive Webapplikationen zu bauen. Ebenfalls ermöglicht React einen modularen Aufbau, wodurch die Wiederverwendung der einzelnen Frontend-Komponenten einfacher möglich ist. Ergänzend dazu wird das Frontend-CSS-Framework Boostrap benutzt.

Zum Anzeigen eines PDF-Dokuments im Browser wird PDF.js verwendet, welches das Einbetten der PDF-Datei in HTML vereinfacht. Der Webserver wird mit Hilfe von Node.js betrieben.

Web Server	Node.js
Content API	React.js
Style	Bootstrap
PDF Rendering Library	Pdf.js

Backend

Das Backend entspricht einer REST-API, welche an eine relationelle Datenbank angeschlossen ist. Die REST-API wird dabei durch FastAPI spezifiziert (und automatisch dokumentiert), und per Uvicorn als ASGI (*Asynchronous Server Gateway Interface*) Server instanziiert. Zur Definition des Datenmodells (ORM) und möglicher Migration wird SQLAlchemy eingesetzt, welches sich mit einer SQLite Datenbankinstanz verbindet. Zum JSON Parsing und Übertragen in ORM wird Pydantic eingesetzt. Für Echtzeitkommunikation (siehe LiveChat Feature) wurde das Python Packet Websockets eingesetzt.

REST API	FastAPI
ASGI	Uvicorn
DBMS	SQLite
ORM	SQLAlchemy
Parsing	Pydantic
Echtzeitkommunikation	Websockets
Datenbank Migrationstool	Alembic

Weitere Tools, deren Anwendungsgebiet weniger relevant waren, sind zur Vollständigkeit im Abschnitt *Lizenz* gelistet.

Versionsverwaltung/CI

1. GitHub
2. GitHub Actions

Wie von dem Unternehmen vorgegeben, wird als Versionsverwaltungstool GitHub verwendet.
Im Laufe des Projekts haben wir uns noch Gedanken über eine CI/CD-Pipeline gemacht. Wir haben uns für GitHub Actions als CI/CD-Tool entschieden, was naheliegend war, da das Projekt bereits auf GitHub gehostet wird.

Kommunikation

1. Wrike (Online-Tool für Projektmanagement)
2. Microsoft Teams
3. Discord

Wrike haben wir auf Vorschlag von unserem Auftraggeber benutzt, um abseits der AG-Treffen miteinander kommunizeren zu können und Anforderungen schriftlich festhalten zu können.

Für die alle zwei Wochen stattfindenden AG-Treffen wurde Microsoft Teams verwendet. Hierbei war vor allem das Screensharing sehr nützlich, um Herrn Dr. Nofer die bisherigen Fortschritte präsentieren zu können.

Innerhalb des Teams haben wir auf einem eigenen Discord-Server kommuninizert.

1.3. Architektur

Das Projekt ist in ein Frontend (React.js auf node server) und ein Backend (FastAPI) aufgeteilt, die mithilfe von RESTful Services kommunizieren. Im Frontend kann man auf verschiedene Unterseiten navigieren. Die verschiedenen Komponenten stellen autonom HTTP-Anfragen an das Backend und erhalten Daten vom Server zurück, wobei sowohl Anfrage als auch Antwort im JSON-Format gehalten sind. Die wesentlichen Komponenten des Backends, welches primär Nutzer*innen und Jobdaten verteilt und speichert, sind die sogenannten Router, Controller und Datenbankmodelle. Die Modelle definieren, wie die Datenbanktabellen aussehen, die Router sind für die Verarbeitung der HTTP-Anfragen verantwortlich, und die Controller sind eine Zwischenebene zwischen diesen beiden Komponenten. Es handelt sich also nicht um ein reines Model-View-Controller-Entwurfsmuster, denn ”View” (entspricht Router) hat gar keinen direkten Zugriff auf das Model, sondern nur über die Controller. Stattdessen handelt es sich hier um Model-View-Presenter (Presenter entspricht dem, was bei uns Controller heißt). Die Schnittstelle zur Datenbank bildet SQLAlchemy, ein Object-relational mapping Tool für Python. Für Object-relational mapping haben wir uns wegen sicherheitstechnischen Aspekten entschieden.

1.4. Vision

In Zukunft soll die Anbindung an ein oder mehrere ATS (engl. Applicant Tracking System - Bewerbermanagementsystem), wie beispielsweise *SAP Success Factors*, eingebaut werden. Dieses Ziel wurde auch bei uns am Anfang definiert, konnte jedoch wegen Unklarheiten bezüglich Softwarelizenzen sowie Kosten des ATS und zugehörigen APIs nicht aufgegriffen werden. Nachfragen eines Mitarbeiters des Auftraggebers bei SAP Success Factors blieben nach unserem aktuellen Kenntnisstand unbeantwortet.
Die Software soll in der Zukunft um weitere Wizards erweitert werden, die jeweils eine Bewertung der Kandidat*innen ergeben. Denkbar wäre auch ein selbst implementierter Video-Chat oder eine Anbindung an Dienste wie Zoom oder Microsoft Teams, um Interviews mithilfe einer Webcam durchführen zu können. Das Datenmodell wird in Zukunft möglicherweise verfeinert werden, um zum Beispiel Skills zusammenfassen zu können; aus dieser Gruppe soll dann von Bewerber*innen mindestens ein Skill vorhanden sein (sogenannte OrSkills). Ein Beispiel dafür wären Cloud-Plattformen: Ein*e Kandidat*in soll für die Stellenausschreibung ”Cloud Engineer” Erfahrungen mit *AWS*, *Microsoft Azure* oder *Google Cloud Platform* haben, allerdings nicht zwingend mit all diesen Diensten, sondern nur mit mindestens einem.

2. Qualitätssicherung

2.1. Qualitätsziel 1: Benutzbarkeit

Bezug zum Projekt: Das QS-Ziel Benutzbarkeit ist von großer Bedeutung, da die Software später in mehrere Bewerbermanagementsysteme (ATS) integriert und daher als ergänzendes Tool oft verwendet werden soll. Da der Entscheidungsprozess bei der Berufsbewerbung für die Firmen durch die Software vereinfacht werden soll, indem es Nutzer*innen der Software möglich sein soll unkompliziert verschiedene Bewerber*innen zu bewerten und zu vergleichen, soll die Nutzung der Software also intuitiv und die Einarbeitung minimal sein. Gleichzeitig sollen die Komponenten alle wichtigen Funktionalitäten abdecken.
Um diese Anforderungen zu erfüllen, haben wir uns für das QS-Ziel Benutzbarkeit entschieden.

Maßnahme: Als Maßnahme zur Sicherung der Benutzbarkeit wird eine Nutzungsstudie durchgeführt, um Feedback zum Umgang mit der Software zu erhalten und gegebenenfalls Verbesserungen vorzunehmen.

Prozessbeschreibung: Im Rahmen dieses Projektes war eine Nutzungsstudie mit der Öffentlichkeit nicht möglich, da diese nur zum Gebrauch innerhalb der Firma der AG zur Verfügung steht. Daher wurde die Nutzungsstudie als *Google Form* Fragebogen angelegt und innerhalb der Firma mit Mitarbeiter*innen, die nicht an der Planung beteiligt waren, durchgeführt.
Die Maßnahme wurde am 15.02.2021 mit fünf Teilnehmenden als Qualitative Studie durchgeführt, sodass in einem angemessenen Zeitraum die Änderungsvorschläge und Bugfixes umgesetzt werden konnten.
Zur Durchführung wurden den Teilnehmenden die Fragebögen, mit betreuenden Instruktionen, zur Verfügung gestellt, um das Projekt im Webbrowser zu öffnen. Die Ergebnisse der Studie wurden dann in der Gruppe diskutiert und konkret in User Stories und Tasks unterteilt (Konkret sind User Stories 54 bis 60 erstellt worden). Die Tasks wurden unter den entsprechenden Entwickler*innen aufgeteilt und dann direkt umgesetzt. So konnte das Feedback für eine bessere Benutzbarkeit direkt in unser Projekt eingebaut werden. (siehe Studienprotokoll und Auswertung)

2.2. Qualitätsziel 2: Funktionalität

Bezug zum Projekt: Unser Projekt ist der Grundstein für ein komplexeres Produkt für unseren Auftraggeber (eine IT-Beratung). Das Projekt wird von anderen Mitarbeiter*innen der Beratung erweitert und später auch vertrieben, wobei sich die Geschäftsführung regelmäßig mit der bereits vorhandenen Kundenbasis bezüglich der gewünschten Features des Produkts austauscht. Diese Unternehmen benutzen bereits Bewerbermanagementsysteme und wollen keinen Ersatz, sondern eine Ergänzung dieser Software. Deshalb ist es wichtig, dass unsere Applikation nur die Aspekte abdeckt, die bei den ATS nur mangelhaft oder erst gar nicht vorhanden sind; diese Features müssen aber fehlerfrei und den Kundenanforderungen entsprechend funktionieren.

Maßnahme: Als Maßnahme zur Qualitätssicherung werden Automatische Pytests regelmäßig (während der Entwicklung, als Teil der Code Reviews und in einer CI Pipeline) ausgeführt.

Prozessbeschreibung: Zur Sicherung der Funktionalität verwenden wir unter anderem das Python-Modul pytest. Für die Erweiterbarkeit der Funktionalitäten haben wir das Produkt sehr modular aufgeteilt und benutzen für das Backend die FastAPI, die im Gegensatz zu anderen Frameworks wie Django sehr lightweight ist und somit nur die nötigsten Aufgaben übernimmt. Zu jedem Endpunkt werden in extra User Stories mindestens 4 Tests erstellt (als Anforderung durch die AG mindestens 4), als Teil der Definition of Done (siehe QS Code Review). Später wurde außerdem eine CI/CD Pipeline mithilfe von Github Actions eingerichtet, die bei jeder Pull Request (also eben auch bei den Code Reviews die damit zusammenhängen) automatisch alle Tests ausführt und einen Merge blockiert, sofern die Tests scheitern. Mit dem Plugin pytest-testcov wurde zudem die Testcoverage (also Anteil der passierten Statements im Code) berechnet. Hier sei angemerkt, dass die TestCoverage mit einem Wert von 35% niedrig erscheint. Das ist allerdings der Programmstruktur geschuldet, und in Vereinbarung mit der AG als genügend beurteilt. Genauere Informationen dazu befinden sich in *qs_belege/2_Funktionalität_Massnahme/AutoTest_Gesamt*

2.3. Qualitätsziel 3: Wartbarkeit

Bezug zum Projekt: Das Projekt soll später von anderen Entwickler*innen weitergeführt werden, weswegen ein sauberer Codestil, eine klare Projektstruktur und ein hoher Modularitäts-/Entkopplungsgrad in den bereits vorhandenen Komponenten gegegeben ist. Zudem muss der Code klar kommentiert und dokumentiert werden damit das eventuelle Überarbeiten alter Abschnitte leichter fällt.

Maßnahme: Als Maßnahme zur QS werden für verschiedene Teile verschiedener User Stories Code Reviews vor der Integration in einen zentralen Development Branch der Versionskontrolle durchgeführt.

Prozessbeschreibung: Um dieses Qualitätsziel zu gewährleisten, wurde sich unter anderem auf Best Practices und eine Definition of Done geeinigt. Beispielsweise wurde festgelegt, wie Branches zu erstellen und zu benennen sind oder, dass Code vor einer Pull Request keine oder mindestens gut begründete Warnungen enthalten darf. Alle Konventionen wurden zu Beginn aufgeschrieben und sind während der Entwicklung einzuhalten (diese sind im Folgenden gelistet).
Die Konventionen werden dann kontinuierlich bei jedem Issue mittels Code Reviews geprüft (Code Review Kriterien ebenfalls im Folgenden gelistet). Code Reviews finden immer bei einer Pull Request statt und werden so oft wiederholt, bis alle Kriterien erfüllt sind, anschließend wird der jeweilige Branch gemerged.

2.3.1. Best Practices

Folgende Punkte wurden zu Beginn als Best Practices indentifiziert und festgehalten, sodass in jedem Bereich das Vorgehen klar definiert ist.

Allgemein

- **Branches**
  - Pro Sprint ein Branch

  - Pro Task darauf ein Branch
  - Nach der jeweiligen User Story/ Task benennen
- **Tasks**
  - Nummerierung in aufsteigender Arbeitsreihenfolge
  - Relevante Contributer markieren und mit User Story verlinken
  - Sobald zu einer Task die Pull Request gestellt wurde, Task in review schieben und die pullrequest in der task verlinken
- **Kommunikation**
  - Diskussionen, welche direkt mit bestehenden Issues zusammenhängen, sollten in den Kommentaren dieser geführt werden.
  - Fragen zu bestehendem Code/zur Software (aber nicht direkt zu einem Issue) sollten als neues Issue erstellt werden. Dieses sollte dann nicht mit dem Projekt oder dem Milestone verknüpft werden. Dabei bitte nicht vergessen, einen bestimmten Nutzer oder die gesamte Gruppe zu taggen, damit das Issue auch gesehen wird.
- **Nomenklatur**
  - Alle Namen - besonders in Code - sind auf Englisch
  - Branches zu Issues werden mit der Issue-Id begonnen, also zum Beispiel: *5-implementingoauth2-scopes*
  - Branches ohne Issues werden in feature/docs/bugfix/... unterteilt, also zum Beispiel: *docsdocumenting-auth-path* oder *feature-implementing-oauth2-scopes*
  - Namen sollten so deskriptiv wie möglich, aber trotzdem kurz sein
  - Gerne auf PyCharm hören, wenn es wegen Benennung "nörgelt"
- **Struktur**
  - Programmfunktionen sollten sinnvoll in Pakete und Unterpakete verteilt sein.

Frontend

- **Allgemein**
  - Barrierefreies Webdesign
  - Responsive Design
- **Frameworks & Libraries**
  - React - UI Library
  - Bootstrap - CSS Framework
  - axios - Senden von HTTP Requests
  - react-router-dom - Routing

Backend

- **Python**
  - Da, wo es geht, Datentypen angeben, um Typen-Fehler zu vermeiden.
  - Python-Dateien, die nur eine Klasse beinhalten, sollten mit CamelCase (und Großbuchstaben am Anfang) benannt werden, zum Beispiel: UserJob.py
  - Allgemein sollte man sich an PEP-8 orientieren, was den Code-Style betrifft (https://www.python.org/dev/peps/pep-0008/)
- **Logging**
  - Python Logging
- **FastAPI**
  - Zugriff auf <XYZ> url/<XYZ>/
  - Create: POST url/<XYZ>/
  - Edit: PUT url/<XYZ>/<ID>
  - Delete: DELETE url/<XYZ>/<ID>
  - View all: GET url/<XYZ>/all
  - View specific: GET url/<XYZ>/<ID>
- **Python HTTP**
  - Für HTTP Fehlermeldungen sollten ausschließlich Exceptions aus dem Paket *backend.exceptions.http* verwendet werden.
  - Für den Rückgabewert *204 No Content* soll der *no_content* Decorator verwendet werden, um ein fehlendes Feature zu ersetzen.

2.3.2. Definition Of Done

Um das Erstellen von Tasks und Pull Requests einheitlich zu gestalten, sind folgende Punkte in der Definition of Done definiert worden:

Pull Request

- Code muss vollständig dokumentiert sein
- Code darf keine Warnungen enthalten, die nicht gut begründet sind
- Code darf nur ToDo's enthalten, die für andere Entwickler relevant sind und nicht bereits in dem Task erledigt werden können
- Jede REST-Schittstelle muss mit mindestens 4 verschiedenen Anfragen durch pytest getestet werden
- Das Frontend muss eigenständig ohne, mit oder mit defekter API funktionieren und den*die Nutzer*in über Fehler informieren

Task Erstellung

- Ein Task soll eine in sich möglichst geschlossene Einheit sein
- Ein Task ist so ausführlich beschrieben, dass jemand Fachkundiges, der*die sich vorher nicht mit dem Code beschäftigt hat, weiß, was zu tun ist
- Ein Task soll einen kleinen Teil einer User Story darstellen
- Ein Task soll Akzeptanzkriterien beinhalten, die möglichst den Teil der Akzeptanzkriterien der User Story abbilden, der von dem Task behandelt wird

2.3.3. Review Kriterien

Im Folgenden sind die Punkte der Checkliste für unsere Code Reviews gelistet. Die Checkliste ist als Template für alle Code Reviews verwendet worden und wurde nicht erweitert.

- Definition Of Done
  - Code vollständig dokumentiert?
  - Keine Warnungen im Code?
  - Keine unnötigen Todos im Code?
- Best Practices
  - Namen im Code in englischer Sprache?
  - Branch/Issue/PR Name sinnvoll?
  - Namen deskriptiv und kurz?
  - Variablen sinnvoll benannt?
  - Variablen Identifier eindeutig?
  - Sinnvolle Programmstruktur?
- Backend
  - Python PEP-8 erfüllt?
  - Unnötige/circular Imports?
  - Weitere Warnungen?
  - Error Codes bei fehlerhaften Requests?
  - Datentypen für Query- und URL-Parameter spezifiziert?
  - JSON-Eingabemodell spezifiziert?
  - (Bei Delete o.Ä) no content Response spezifiziert?
  - Single Class CamelCase?
- Tests für Backend
  - Mindestens 4 für den gesamten Endpunkt?
  - Alle Tests werden erfolgreich abgeschlossen?
  - Edge Case Fehler in manuellen Tests? (500 bei falschen Eingabeparametern)

- Frontend
  - Fehler bei unerreichbarem Backend gefunden?
  - Design barrierefrei?
  - Design responsive?
  - Modularer Aufbau mittels React Components?
  - Sonstige Auffälligkeiten?
- Akzeptanzkriterien
- Ergebnis
- Passed?
- Consequences

3. Abweichende QS Ziele

Alle genannten QS Ziele konnten wie geplant durchgeführt und durch Maßnahmen gesichert werden.

4. Projektverlauf und Projektgefährdende Ereignisse

Da unsere AG ursprünglich zwei Projekte angeboten hatte (“40: Transparent Skills Alignment Mechanism - CV Interview Wizard” und “41: Transparent Skills Alignment Mechanism - Decision Heatmap”), aber nur für Projekt 40 uns als Gruppe bekommen hat, wurden die Themen beider Projekte von der AG kombiniert.
Das bedeuted wir haben beide Themen in einem Projekt, mit jeweils angepassten / geringeren Anforderungen, bearbeitet.
Wir wollten das hier nur kurz erwähnen, damit keine Verwirrung aufkommt, weshalb das Thema “Decision Heatmap” hier mit in dem Projekt bearbeitet wurde.

Projektverlauf: Das erste Planungstreffen mit dem Auftraggeber fand am 25.11.2020 statt. Die Ergebnisse des Projekts wurden am 17.03.2021 im finalen Treffen übergeben.
Im Verlauf des Agile Prozesses fanden im 2-Wochen-Turnus jeweils ein Review- als auch ein Sprint-Meeting mit dem AG statt.

- Iteration 1: 25.11.2020 – 09.12.2020
- Iteration 2: 09.12.2020 – 23.12.2020
- Iteration 3: 23.12.2020 – 06.01.2021
- Iteration 4: 06.01.2021 – 20.01.2021
- Iteration 5: 20.01.2021 – 03.02.2021
- Iteration 6: 03.02.2021 – 17.02.2021
- Iteration 7: 17.02.2021 – 03.03.2021
- Iteration 8: 03.03.2021 – 17.03.2021 (Abgabe an AG)
- Iteration 9: 17.03.2021 – 31.03.2021 (Ausschließlich Dokumentation)

Projektbeginn: Da unser Projekt den Grundstein für eine sehr große, sich weiterentwickelnde Webapplikation mit kommerzieller Verwendung darstellt, wurde von Seiten der AG ein Grundgerüst für das System zur Verfügung gestellt. Nils Richter (Praktikant bei unserem Auftraggeber) hat als Systemadmin den Demonstrations-LiveServer sowie das Git Repository der Firma verwaltet und uns am Anfang in den bestehenden Teil eingearbeitet. Außerdem hat er als zusätzlicher Reviewer in der ersten und zweiten Iteration den Review Prozess begleitet und uns die Vorgaben der AG bezüglich Codequalität detaillierter vermittelt (siehe Maßnahme *Wartbarkeit*). Somit konnte garantiert werden, dass der neue Quellcode von uns sinnvoll mit dem Fundament des Projektes zusammenarbeitet und die Vorgaben bezüglich Codequalität, Testing und Dokumentation der bisherigen Firmenstruktur der AG folgen.
Vor unserem Entwicklungsbeginn hat das Projekt bereits im Backend ein rudimentäres AccountSystem mit Authentifikation und Dateiupload sowie ein Frontend mit Login und Dateiupload Formular beinhaltet.

Projektabschluss: Nach der finalen Übergabe des Projektes führte unser Team die Entwickler, welche (Stand 17.03.2021) das Projekt übernommen haben, in den Aufbau des Projektes ein.

Abweichendes Zeitbudget: Da sich für uns zu dem Abschluss der Entwicklungszeit hin noch ein paar Änderungen im Datenmodell als sinnvoll herausstellten, haben wir mit Absprache der AG die neuen Änderungen freiwillig implementiert. Dadurch kam es zu einer deutlichen Überschreitung des erwarteten Zeitaufwands für das Projekt im Backend und vorallem auch im Frontend, da ein paar Komponenten modifiziert werden mussten, sodass diese mit dem neuen Datenmodell weiter funktionieren.
Ebenfalls haben wir aus Interesse am Ende noch den Chat zu einem Live-Chat erweitert, den wir als zusätzliches Feature freiwillig noch implementieren wollten, welcher aber dadurch auch Einfluss auf die Abweichung der Zeiten hatte.

Projektgefährdende Ereignisse: Im Laufe des Projekts gab es keine projektgefährdenden Ereignisse.
Aufgrund seiner (dem Orga-Team bekannten) chronischen Erkrankung war Teammitglied Philip für mehrere Abschnitte des Projektes für einzelne Tage nicht verfügbar, jedoch konnte die Arbeit von anderen Teammitgliedern übernommen werden und es konnten so trotzdem die Anforderungen problemlos erfüllt werden.

5. Softwarelizenz

Der für dieses Projekt erstellte Code steht im Einvernehmen aller Beteiligten unter folgender Lizenz:

Das Projekt, “as-is” bestehend aus dem Quelltext, Dokumentation, Spezifikationen und Testabläufen ist intellektuelles Eigentum der DIErektBERATUNG AG (Lizenz: Commercial). Sämtliche Open Source Komponenten, die (Stand 17.03.2021) eingesetzt wurden, sind anschließend gelistet.

- Frontend
    1. axios
    2. bootstrap
    3. jest-dom
    4. NodeJS
    5. PDF.js
    6. prop-types
    7. react
    8. react-bootstrap
    9. react-color
    10. react-dom
    11. react-router-dom
    12. react-scripts
    13. user-event
    14. web-vitals
- Backend
    1. aiofiles
    2. alembic
    3. atomicwrites
    4. attrs
    5. certifi
    6. chardet
    7. click
    8. colorama
    9. fastapi

10. h11
11. idna
12. iniconfig
13. Mako
14. MarkupSafe
15. packaging
16. pluggy
17. py
18. pydantic
19. pyparsing
20. pytest
21. pylint
22. python-dateutil
23. python-editor
24. python-multipart
25. requests
26. six
27. SQLAlchemy
28. starlette
29. toml
30. urllib3
31. websockets
32. uvicorn

Die Teilnehmer*innen des Projektes haben sich vertraglich zur Verschwiegenheit bezüglich des Projektes gegenüber dritten Personen verpflichtet (siehe Dokument *nda_tsam.pdf*).

A. Anhang

A.1. Abkürzungsverzeichnis

Abkürzungen: Im Folgenden sind diverse Abkürzungen berschrieben, welche in der Projektdokumentation und den User Stories enthalten sind.

- Allgemeine Abkürzungen
  - ATS - Applicant Tracking System (Bewerbermanagementsystem)
  - ASGI - AsynchronousServer Gateway Interface
  - DBMS - Datenbankmanagementsystem
  - ORM - Object Relational Mapping
  - CI/CD - Continuous Integration / Continuous Deployment
  - AWS - Amazon Web Services
  - NDA - Non-Disclosure Agreement
  - CV - Curriculum Vitae (Lebenslauf)
- Projektspezifische Abkürzungen
  - CSG - Candidate Skill Grade
  - GHF - Grade Heat Function
  - GSF - Grade Scale Function
  - WSF - Weight Scale Function

A.2. Stand des Projekts zur Abgabe / Datenmodell

Das Projekt wurde bis zur Abgabe noch um viele Funktionalitäten erweitert, weshalb das Datenmodell und entsprechend das ER-Diagramm zum Ende hin sehr groß wurde.
Die Grundkomponenten sind gleich geblieben bzw. nur leicht verändert worden: Es gibt weiterhin Jobs, User, Candidates, Essential Skills und Interview Questions. Allerdings soll ein*e Kandidat*in pro Essential Skill nicht nur eine Bewertung, sondern mehrere haben können. Bewertungen (”Candidate Skill Grades”) sollen außerdem mit einer sogenannten Source assoziiert sein: Eine Source ist jede Instanz, die eine Bewertung vergeben kann. Sources können also z.B. sowohl Interviews, in denen Skills abgefragt werden, aber auch automatisierte Online-Assessments sein, die möglicherweise auch noch von einem externen Anbieter stammen (z.B. Codility für Software-Jobs). Eine Source ist immer genau einem Essential Skill zugeordnet, und genau einem Assessment (welches letztendlich nur eine Gruppierung von Sources ist).
CV Wizard und Interview Wizard wurden zu einem Wizard zusammengefasst, bei dem zwei Chat-Fenster und ein PDF-Viewer, in dem man Markierungen einfügen kann, vorhanden ist (sog. DocsComponent). Das erste,

obere Chat-Fenster bezieht sich immer auf eine Source; für jede Source existiert ein eigener Chat, den man auswählen kann. Markierungen im CV sind weiterhin möglich. Neu hinzugekommen sind Observations: Nachrichten bzw. Bemerkungen, die für den*die Bewerber*in unsichtbar sind, aber sichtbar für Interviewer*innen und andere Mitarbeitende (zusammengefasst als Observer) eines Unternehmens. Das untere Chat-Fenster enthält einen speziellen Chat, der dafür da ist, je eine bestimmte Interview-Frage beantworten zu können.

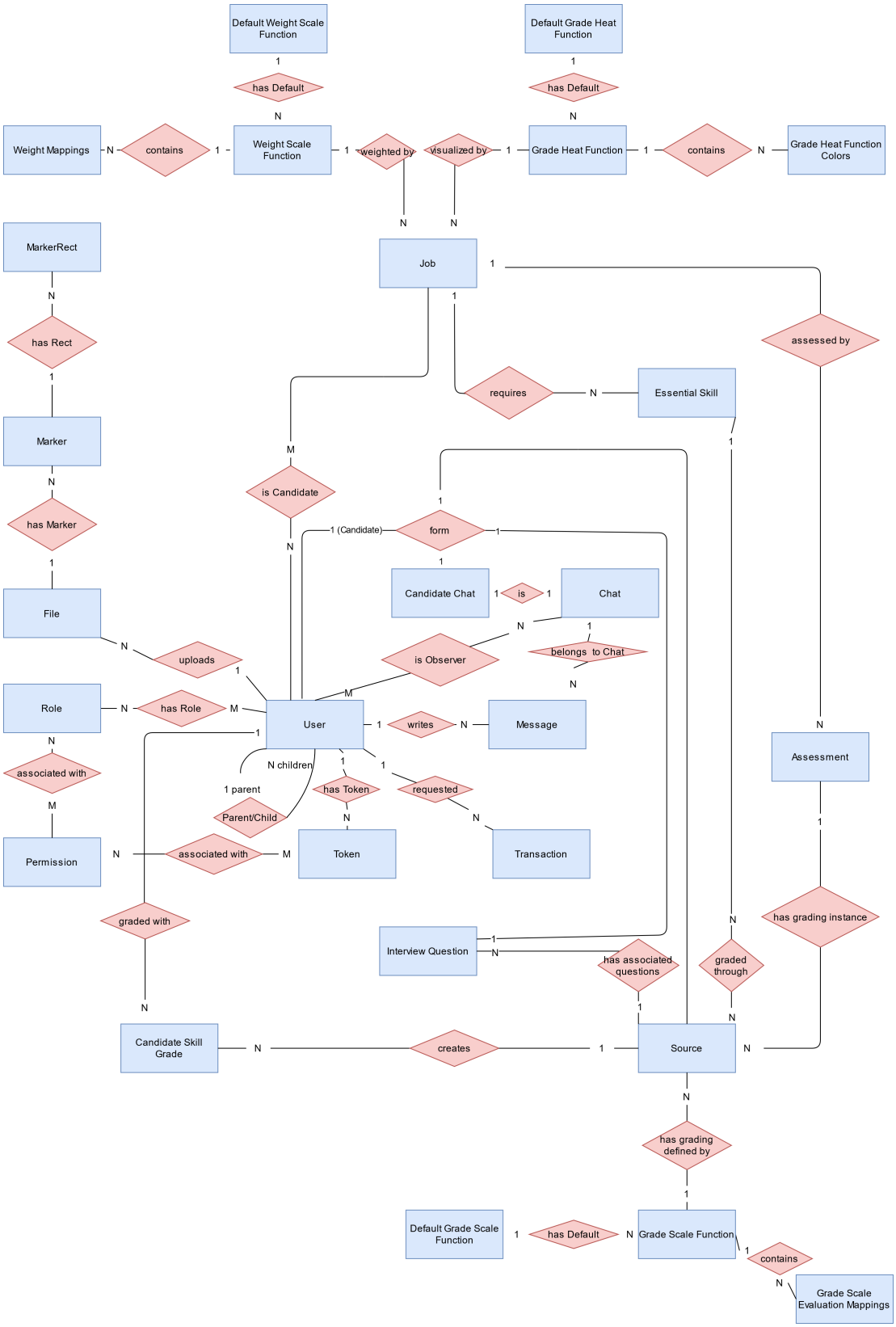

Abbildung A.1.: ER Diagramm des Datenmodells