

Inhaltsverzeichnis

[1 Anforderungsanalyse 3](#_Toc76118181)

[2 Vorgehen und Meilensteine 3](#_Toc76118183)

[3 Use-Case-Diagramm 4](#_Toc76118184)

[4 Klassendiagramm 7](#_Toc76118198)

[5 ER-Diagramm 14](#_Toc76118205)

[6 Installation der fertigen Anwendung 15](#_Toc76118206)

[7 Nutzerhandbuch 19](#_Toc76118210)

# 1 Anforderungsanalyse

## Zielsystem

Die zu erstellende Anwendung soll Studenten die Möglichkeit bieten, Lernpartner zu finden. Dazu muss jede Person einen Account anlegen und ihre Lernvorlieben angeben. Über diese Lernvorlieben kann ein entsprechender Algorithmus dann passende Lernpartner vorschlagen. Diese Vorschläge umfassen sowohl Personen als auch Lerngruppen. Diesen Vorschlägen können dann Kontaktanfragen gesendet werden. Werden diese akzeptiert, können die Nutzer daraufhin über einen Chat miteinander kommunizieren. Anfragen, die an Gruppen gestellt werden, können von jedem Gruppenmitglied angenommen oder abgelehnt werden. Weiterhin können die Accountdaten bearbeitet und der gesamte Account gelöscht werden.

Die Authentifizierung erfolgt über Google Firebase und die Anwendung soll über Google Cloud gehostet werden. Die App verfügt über eine Clientseite, geschrieben mit dem Framework React und eine Serverseite, die das Python Webframework Flask nutzt. Client und Server kommunizieren über eine REST-API. Die Daten werden in einer MySQL-Datenbank gespeichert und von dort aus abgerufen.

# 2 Vorgehen und Meilensteine

Im Folgenden werden die Meilensteine kurz erläutert.

Für den ersten Meilenstein galt es, sich in einer Gruppe aus 5-6 Personen zu finden und diese Zusammenstellung anzugeben. Die vorliegende Anwendung wurde von sechs Personen erstellt, deren Namen auf dem Deckblatt genannt sind.

Der zweite Meilenstein umfasste die Erstellung des Use-Case-Diagramms, das die verschiedenen Möglichkeiten abbildet, wie die Anwendung genutzt werden kann. Das aktuelle und angepasste Use-Case-Diagramm befindet sich unter 3.

Im Rahmen des dritten Meilensteins wurde dann das Klassen- und das Entity-Relationship-Diagramm erstellt. Diese repräsentieren die verwendeten Klassen der Anwendung und den Aufbau der Datenbank. Im Laufe der Entwicklung wurden beide Diagramme mehrmals angepasst und überarbeitet. Die aktuellen Versionen sind unter 4., beziehungsweise 5. Abgebildet.

Der vierte Meilensteil besteht aus der Abgabe des Peer-Feedbacks, das dazu dient, die eigenen Gruppenmitglieder hinsichtlich Mitwirkung, um Team einzuschätzen. Es dient vor allem dazu, den Gruppenmitgliedern Rückmeldung zu ihrer Performance zu geben und Probleme frühzeitig zu erkennen.

Der fünfte Meilenstein ist die Abgabe des vollständigen Source-Codes der Anwendung, sowie diese Dokumentation. Der Source-Code soll mit Kommentaren entsprechend umfangreich dokumentiert sein.

Die beiden letzten Meilensteine umfassen die Präsentation fertigen Zielsystems in der Cloud und das individuelle Fachgespräch. Diese werden hier in der Dokumentation nicht behandelt.

## 3 Use-Case-Diagramm

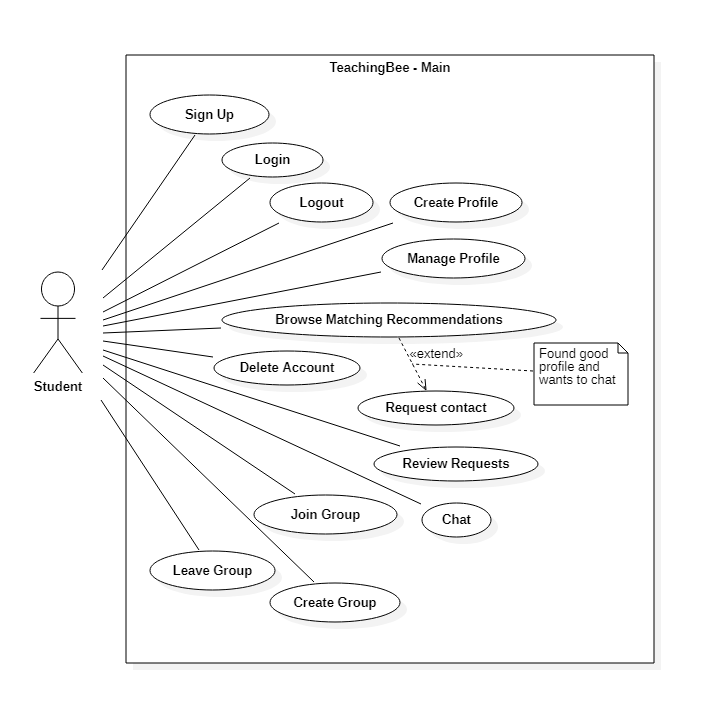


Abbildung 1: Use Case Diagramm. Eigene Darstellung.

In Abbildung 1 ist das aktuelle Use Case Diagramm des Programms zu sehen. Es kennt einen Akteur, nämlich den *Student*. Der *Student* kennt insgesamt 12 Use Cases mit einer Extension.

Abbildung 2: Use Case Diagramm

## Login

Der Nutzer kann sich über FireBase einloggen. Die Aufforderung dazu erscheint automatisch, wenn kein FireBase-Token in den Cookies des Nutzers gefunden wird.

## SignUp

Wenn ein Nutzer sich zum ersten Mal einloggt, wird seine FireBase-ID nicht in der Datenbank gefunden. Ist dies der Fall, wird er dazu aufgefordert, einen Account zu erstellen. Dabei muss er persönliche Daten angeben, wie zum Beispiel seinen Vor- und Nachnamen, seinen Studiengang oder sein Geburtsdatum. Diese Daten werden dann zusammen mit seiner FireBase-ID in der Datenbank abgespeichert.

## Logout

Wenn der Nutzer sich ausloggen will, muss er nur den entsprechenden Button anklicken. Er wird dann automatisch von FireBase ausgeloggt. Will er den Dienst wieder nutzen, muss er sich erneut einloggen.

## Create Profile

Hat sich ein Nutzer neu angemeldet, muss er noch sein Profil erstellen. Er wird jedes Mal dazu aufgefordert, wenn er die App startet, ohne das ein Profil vorhanden ist. Um sein Profil zu erstellen, muss er ein paar Fragen zu seinem Lernverhalten und seinen Interessen beantworten. Die Antworten werden dann in der Datenbank abgespeichert und dienen später dazu, beim Matching möglichst relevante Vorschläge zu machen.

## Manage Profile

Will der User sein Profil anpassen, kann er auf den entsprechenden Button klicken. Daraufhin hat er die Möglichkeit, die Werte anzupassen und die alten Daten zu überschreiben.

## Browse Matching Recommendations:

Will ein Nutzer neue Lernpartner oder Lerngruppen finden, klickt er auf den Matching Button. Er bekommt dann eine Liste mit den relevantesten Studenten und Gruppen angezeigt.

### Request Contact

Hat der Nutzer beim Browser der Recommendations ein Profil oder eine Gruppe gefunden, zu der er Kontakt aufnehmen möchte, kann er eine Anfrage senden. Die Anfrage wird dann dem entsprechenden Nutzer in der Liste der Anfragen angezeigt.

## Delete Account

Der Nutzer hat die Möglichkeit seinen Account zu löschen. Die Einträge werden aus der Datenbank entfernt und wenn der Nutzer die Webseite wieder öffnet, muss er erneut durch den Prozess der Anmeldung.

## Review Requests

Der Nutzer kann eine Liste mit Anfragen öffnen und entscheiden, ob er sie annehmen möchte. Nimmt er eine Anfrage an, taucht der andere Nutzer in der Chatliste auf.

## Chat

Hier findet der Nutzer sämtliche Kontakte, die seine Anfrage angenommen haben, dessen Anfragen er angenommen hat, sowie Gruppen, in denen er Mitglied ist. Er kann einen Eintrag wählen, um ein Chatfenster zu öffnen und Nachrichten zu lesen und zu schreiben.

## Join Group

Der Nutzer kann einer Gruppe beitreten, die er durch das Matching gefunden hat. Er kann dann mit der Gruppe chatten und Termine zum Lernen vereinbaren.

## Create Group

Wenn ein Nutzer eine Gruppe erstellen will, muss er nur den entsprechenden Button anklicken. Im Anschluss öffnet sich ein Pop-up, in dem er den Gruppenname und die Gruppenbeschreibung erstellen kann. Die Gruppe wird nach ihrer Erstellung automatisch in der Chat-Liste angezeigt.

## Leave Group

Wenn der Nutzer einer Gruppe verlassen will, muss er nur den entsprechenden Button anklicken. Seine Gruppenzugehörigkeit wird aus der Datenbank entfernt. Falls der Nutzer als letztes die Gruppe verlässt, wird die Gruppe aus der Datenbank entfernt, zusammen mit all ihren Nachrichten.

# 4 Klassendiagramm

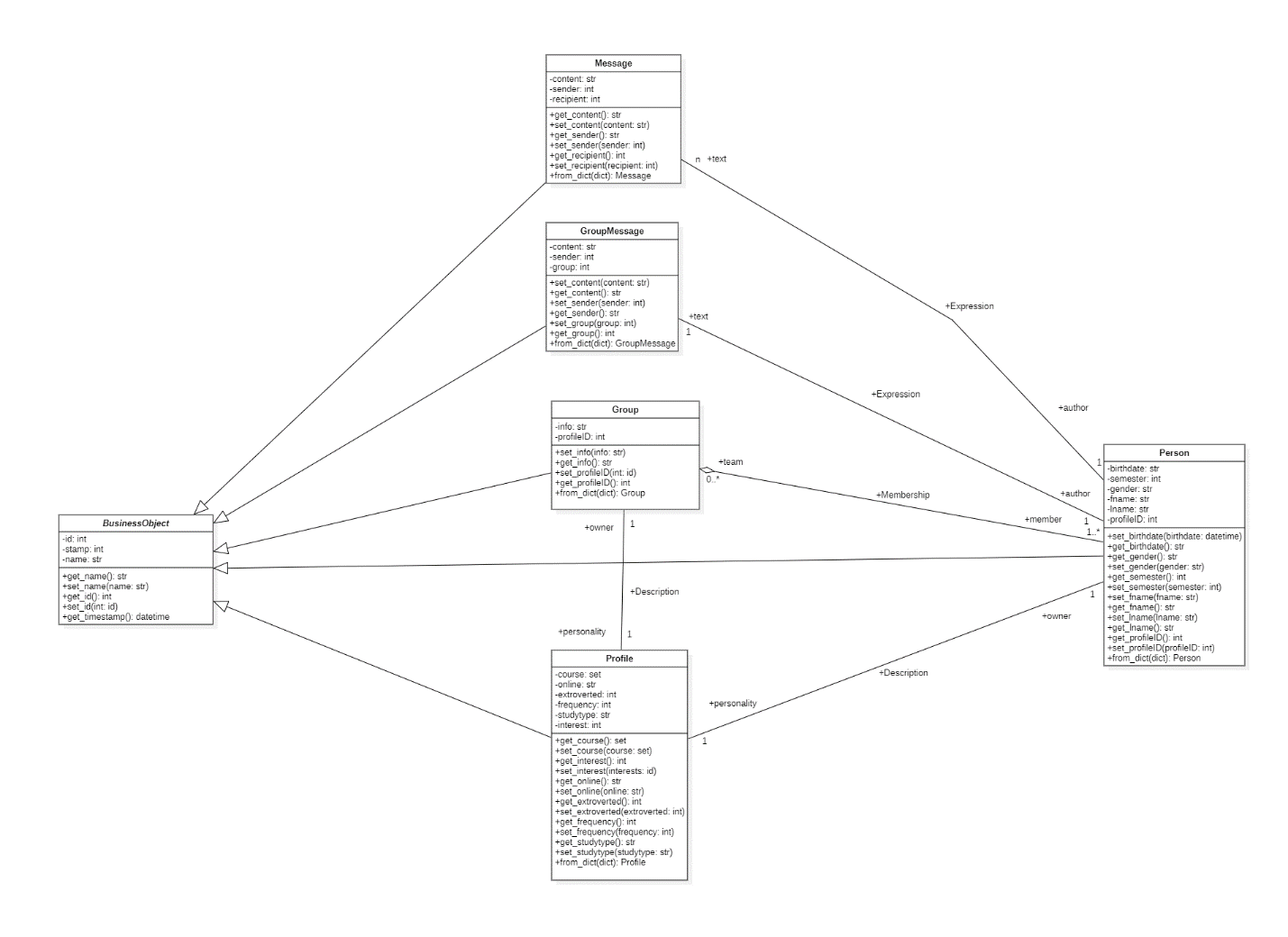


Abbildung 3: Klassendiagramm. Eigene Darstellung.

## BusinessObject:

Die BusinessObject Klasse ist eine Abstrakte Klasse, von der sämtliche anderen Klassen erben. Sie enthält lediglich drei Variablen, nämlich die ID, die Erstellungszeit und einen Namen, sowie die zugehörigen Getter und Setter.

**Attribute:**

*BusinessObject.*\_id

Enthält die ID des Objekts. Die ID wird von der Datenbank beim ersten Schreiben des Objektes vergeben und danach immer mit ausgelesen.

*BusinessObject.*\_stamp

Enthält den Zeitpunkt der Erstellung des Objekts. In der Praxis wird dieser Zeitpunkt beim ersten Schreiben des Objekts in die Datenbank festgehalten und danach immer mit ausgelesen.

*BusinessObject.*\_name

Enthält den Namen des Objektes. Diese Variable wurde bei diesem Projekt nur bei der Klasse Group genutzt.

**Methoden:**

*BusinessObject.*get\_id()

Gibt die ID des Objektes aus.

*BusinessObject.*set\_id(value)

Schreibt die ID.

*BusinessObject.*get\_stamp()

Gibt die Erstellungszeit aus.

*BusinessObject.*set\_stamp(value)

Schreibt die Erstellungszeit.

*BusinessObject.*get\_name()

Gibt den Namen aus.

*BusinessObject.*set\_name(value)

Schreibt den Namen.

## Message:

Die Message Klasse enthält jeweils eine Textnachricht mit einem Inhalt, dem Nachrichtentext, sowie dem Absender und Empfänger, die mit ihrer ID gespeichert sind. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject. Daher hat sie auch eine ID und einen Erstellungszeitpunkt.

**Attribute:**

*Message.*\_content

Enthält den Inhalt der Nachricht als Text.

*Message*.\_sender

Enthält die ID des Absenders.

*Message*.\_recipient

Enthält die ID des Empfängers.

**Methoden:**

*Message*.set\_content(content)

Schreibt den Nachrichteninhalt.

*Message*.get\_content()

Gibt den Nachrichteninhalt aus.

*Message*.set\_sender(sender)

Schreibt die ID des Absender.

*Message*.get\_sender()

Gibt die ID des Absenders aus.

*Message*.set\_recipient(recipient)

Schreibt die ID des Empfängers.

*Message*.get\_recipient()

Gibt die ID des Empfängers aus.

*Message*.\_\_str\_\_()

Gibt alle Variablen der Klasse als ein kompakten String aus.

*Message*.from\_dict(dictionary)

Die from\_dict Methode ist eine statische Methode. Nimmt einen Dictionary mit den Keys *content*, *sender*, *recipient* an und erstellt daraus ein Message Objekt.

## GroupMessage

Die GroupMessage Klasse ist beinahe identisch mit der Message Klasse. Sie unterscheidet sich größtenteils in ihrem Sinn und ihrer Verwendung. Sie enthält jeweils eine Textnachricht mit einem Inhalt, dem Nachrichtentext, sowie dem Absender und der jeweiligen Gruppe, die mit ihrer ID gespeichert sind. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject. Daher hat sie auch eine ID und einen Erstellungszeitpunkt.

**Attribute:**

*GroupMessage.*\_content

Enthält den Inhalt der Nachricht als Text.

*GroupMessage*.\_sender

Enthält die ID des Absenders.

*GroupMessage*.\_group

Enthält die ID der Gruppe, die die Nachricht empfängt.

**Methoden:**

*GroupMessage*.set\_content(content)

Schreibt den Nachrichteninhalt.

*GroupMessage*.get\_content()

Gibt den Nachrichteninhalt aus.

*GroupMessage*.set\_sender(sender)

Schreibt die ID des Absenders.

*GroupMessage*.get\_sender()

Gibt die ID des Absenders aus.

*GroupMessage*.set\_group(group)

Schreibt die ID des Empfängers.

*GroupMessage*.get\_group()

Gibt die ID des Empfängers aus.

*GroupMessage*.\_\_str\_\_()

Gibt alle Variablen der Klasse als ein kompakten String aus.

*GroupMessage*.from\_dict(dictionary)

Die from\_dict Methode ist eine statische Methode. Nimmt einen Dictionary mit den Keys *content*, *sender*, g*roup* an und erstellt daraus ein GroupMessage Objekt.

## Group

Die Group Klasse hält Lerngruppen. Sie enthält Gruppenbeschreibung als Text und die ID des zugehörigen Gruppenprofils. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject. Daher hat sie auch eine ID, einen Namen und einen Erstellungszeitpunkt.

**Attribute:**

*Group*.\_info

Enthält die Gruppenbeschreibung als String.

*Group*.\_profileID

Enthält die ID des zugehörigen Gruppenprofils.

**Methoden:**

*Group*.set\_info(info)

Schreibt die Gruppenbeschreibung.

*Group*.get\_info

Gibt die Gruppenbeschreibung aus.

*Group*.set\_profileID(id)

Schreibt die Profil-ID.

*Group*.get\_profileID()

Gibt die Profil-ID aus.

*Group*.\_\_str\_\_()

Gibt alle Variablen der Klasse als ein kompakten String aus.

*Group*.from\_dict(dictionary)

Die from\_dict Methode ist eine statische Methode. Nimmt einen Dictionary mit den Keys *id*, *name*, *info* und *profileID* an und erstellt daraus ein Group Objekt.

## Profile

Die Klasse Profile enthält die Lernprofil von Personen und Gruppen. Sie enthält den Studiengang, den Lerntyp, die Extrovertiertheit, die Lernfrequenz, die Präferenz, ob online oder offline gelernt werden soll, sowie die persönlichen Interessen. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject. Daher hat sie auch eine ID und einen Erstellungszeitpunkt.

**Attribute:**

*Profile*.\_course

Enthält den jeweiligen Studiengang.

*Profile*.\_studytype

Enthält den Lerntyp.

*Profile*.\_frequency

Enthält die präferierte Lernfrequenz.

*Profile*.\_online

Enthält die Online-Präferenz.

*Profile*.\_interest

Enthält die Interessen.

**Methoden:**

*Profile*.get\_course()

Gibt den Studiengang aus.

*Profile*.set\_course(value)

Schreibt den Studiengang.

*Profile*.get\_studytype()

Gibt den Lerntyp aus.

*Profile*.set\_studytype(value)

Schreibt den Lerntyp.

*Profile*.get\_extroverted()

Gibt die Extrovertiertheit aus.

*Profile*.set\_extroverted(value)

Schreibt die Extrovertiertheit.

*Profile*.get\_frequency()

Gibt die Lernfrequenz aus.

*Profile*.set\_frequency()

Schreibt die Lernfrequenz.

*Profile*.get\_online()

Gibt die Online-Präferenz aus.

*Profile*.set\_online(value)

Schreibt die Online-Präferenz.

*Profile*.get\_interest()

Gibt die Interessen aus.

*Profile*.set\_interest(value)

Schreibt die Interessen.

*Profile*.\_\_str\_\_()

Gibt alle Variablen der Klasse als ein kompakten String aus.

*Profile*.from\_dict()

Die from\_dict Methode ist eine statische Methode. Nimmt einen Dictionary mit den Keys *id*, *course*, *studytype*, *extroverted,* *frequency* *online* und *interest* an und erstellt daraus ein Message Objekt.

## Person

Die Klasse Person bildet einen Nutzer ab. Sie hält dafür Informationen wie den Vor- und Nachnamen, das Geburtsdatum, das Fachsemester, das Geschlecht sowie einen Verweis auf das zugehörige Profil. Außerdem erbt die Klasse von der abstrakten Klasse BusinessObject. Daher hat sie auch eine ID und einen Erstellungszeitpunkt.

**Attribute:**

*Person*.\_fname

Enthält den Vornamen des Nutzers.

*Person*.\_lname

Enthält den Nachnamen des Nutzers.

*Person*.\_birthdate

Enthält das Geburtsdatum des Nutzers.

*Person*.\_semester

Enthält das Fachsemester des Nutzers.

*Person*.\_profileID

Enthält die ID des zugehörigen Profils.

**Methoden:**

*Person*.get\_fname()

Gibt den Vornamen aus.

*Person*.set\_fname(fname)

Schreibt den Vornamen.

*Person*.get\_lname()

Gibt den Nachnamen aus.

*Person*.set\_lname(lname)

Schreibt den Nachnamen.

*Person*.get\_ birthdate ()

Gibt das Geburtsdatum aus.

*Person*.set\_birthdate(birthdate)

Schreibt das Geburtsdatum.

*Person*.get\_semester()

Gibt das Semester aus.

*Person*.set\_semester ()

Schreibt das Semester.

*Person*.get\_gender()

Gibt das Geschlecht aus.

*Person*.set\_ gender (gender)

Schreibt das Geschlecht.

*Person*.get\_profileID()

Gibt die ID des zugehörigen Profils aus.

*Person*.set\_ profileID (profileID)

Schreibt die ID des zugehörigen Profils.

*Person*.\_\_str\_\_()

Gibt alle Variablen der Klasse als ein kompakten String aus.

*Person*.from\_dict()

Die from\_dict Methode ist eine statische Methode. Nimmt einen Dictionary mit den Keys *id*, *fname*, *lname*, *birthdate*, *semester*, *gender* und *profileID* an und erstellt daraus ein Person Objekt.

# 5 ER-Diagramm

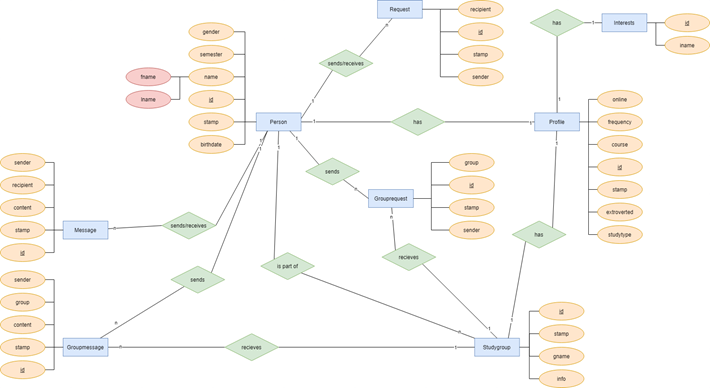


Abbildung 4: ER-Diagramm. Eigene Darstellung.

Eine hochauflösende Version des Bildes befindet sich unter <https://github.com/sihensel/teachingbee/blob/main/docs/ER-diagram.png>

Das ER-Diagramm gibt im Wesentlichen die Struktur des Klassendiagramms wieder, da jede Klasse durch eine Tabelle repräsentiert wird. Darüber hinaus kommen noch folgende Tabellen hinzu:

* Interests (beinhaltet alle zur Auswahl stehenden Interessen, wird mit Profile verknüpft)
* Request (für Kontaktanfragen zwischen Personen)
* Grouprequest (Kontaktanfragen von Personen an Gruppen)

Diese Tabellen haben keine entsprechenden Klassen.

Jede Tabelle besitzt einen Primärschlüssel, der von der Datenbank automatisch vergeben und hochgezählt wird und einen Timestamp, der den Erstellungszeitpunkt des Datensatzes wiedergibt. Innerhalb der Datenbank wurden alle n:m-Beziehungen aufgelöst und alle Relationen von Fremdschlüssel-Beziehungen repräsentiert.

Die Tabellennamen starten mit einem Großbuchstaben, damit sie leichter von Tabellenfeldern zu unterscheiden sind. Einige Tabellen und Felder haben spezielle Namen, um Konflikte mit reservierten Schlüsselbegriffen seitens MySQL zu vermeiden. SO heißt der Timestamp nur *stamp* oder die Tabelle zu Lerngruppen *studygroup* anstatt nur *group*.

Die Erstellung der Datenbank erfolgt über die Datei *dump.sql* im Ordner *database*.

# 6 Installation der fertigen Anwendung

Das GitHub-Repository mit dem Source Code befindet sich unter <https://github.com/sihensel/teachingbee>.

## Voraussetzungen

Um die Anwendung lokal ausführen zu können werden folgende Programme beziehungsweise Laufzeitumgebungen benötigt:

* Python 3.9
* Node.js
* Terminal (wie es etwa in VSCode oder Pycharm eingebunden ist)
* Web-Browser

Darüber hinaus werden einige Python-Module benötigt, die über das Programm pip installiert werden können. Das Anlegen eines Virtual Environments ist empfohlen.

* Flask
* Flask-restx
* Flask-Cors
* mysql-connector-python

Um die benötigten Node-Packages zu installieren, im Terminal in den Ordner */reactfrontend* navigieren und **npm install** ausführen. Welche Module installiert werden, kann in der Datei */reactfrontend/package.json* eingesehen werden.

## Lokal

Um die App lokal zu installieren, muss zuerst die MySQL-Datenbank lokal angelegt werden. Dies kann über ein Tool wie MySQL Workbench oder auch das CLI von MySQL geschehen. Für genauere Hinweise ist die die Dokumentation des Herstellers zu konsultieren. Anschließend müssen in der Datei *Mapper.py* im Ordner */backend/server/db/* noch die entsprechenden Login-Informationen angegeben werden.

Als nächster Schritt muss die Authentifizierung mit Google Firebase eingerichtet werden. Dazu in Firebase ein Projekt für eine Web-App anlegen. In diesem Zug wird automatisch eine Konfiguration erstellt, die dann in die Datei */reactfrontend/firebaseconfig.js* kopiert werden muss. Bei Schwierigkeiten kann die Dokumentation seitens Firebase konsultiert werden.

Danach kann der React-Client kompiliert werden. Dazu im Terminal in den Ordner *reactclient* navigieren und den Befehl **npm run build** ausführen. Das fertige Kompilat steht dann im neuen Order *build* zur Verfügung. Der Inhalt dieses Ordners muss nun nach *backend/static/reactclient* kopiert oder verschoben werden.

Anschließend kann der Flask-Server gestartet werden, indem die Datei ‘main.py’ ausgeführt wird. Die Anwendung steht dann im Browser unter <http://localhost:5000/static/reactclient/index.html> zur Verfügung. Die REST-API befindet sich unter [http://localhost:5000](http://localhost:5000/).

## Cloud

Um die App auf Google Cloud bereit zu stellen, muss zuerst über die Google Cloud Console ein neues Projekt angelegt werden. Danach in Google Cloud eine MySQL-Datenbank erstellen und über die Datei *dump.sql* befüllen. In der Datei */backend/server/db/Mapper.py* wieder die entsprechenden Login-Informationen angeben.

Als nächstes muss die Google App Engine aktiviert und auf dem lokalen Rechner die Google Cloud SDK installiert werden. Hier müssen noch die beiden Komponenten **app-engine-python** und **app-engine-python-extras** installiert und das gerade erstellte Projekt ausgewählt werden. Es wird ausdrücklich auf die Dokumentation des Herstellers verwiesen. Anschließend die Firebase-Konfiguration erstellen und den React-Client kompilieren (siehe oben). Dann kann die Anwendung mit der Google Cloud SDK deployed werden. Dazu in den Ordner backend navigieren und den Befehl **gcloud app deploy** ausführen.

Ist der Prozess abgeschlossen, kann die Anwendung mit **gcloud app browse** genutzt werden.

# 7 Nutzerhandbuch

Die App kann über die folgenden Links online aufgerufen werden:

* <https://teachingbee-179c2.ew.r.appspot.com/>
* <https://teachingbee-179c2.ew.r.appspot.com/static/reactclient/index.html>

Öffnet der Nutzer die App, muss er sich immer erst mit seinem Google Account anmelden.

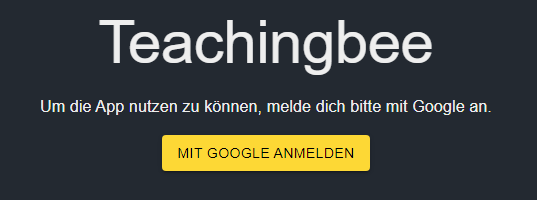


Abbildung 5: Die Anmeldeseite. Eigene Darstellung.

Hat der Nutzer noch keinen Account, muss er sich erst registrieren, in dem die Personendaten und die Lernvorlieben erfasst werden.

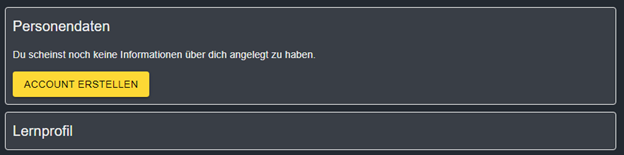


Abbildung 6: Die Accountübersicht. Eigene Darstellung

Ist dies erfolgt, kann er alle Funktionen der Anwendung uneingeschränkt nutzen. In der oberen Navigationsleiste befinden sich die Buttons für jede Funktion.

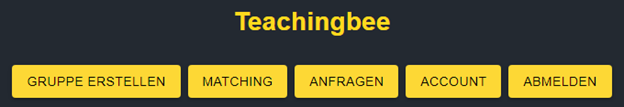


Abbildung 7: Die Navigation-Bar. Eigene Darstellung.

Über den rechten Abmelden-Button kann sich der Nutzer von Firebase abmelden und die App wird geschlossen.

Auf der Startseite werden alle Chats, sowohl mit anderen Personen als auch Gruppenchats, angezeigt. Mit einem Klick auf den jeweiligen Chat werden alle entsprechenden Nachrichten angezeigt.

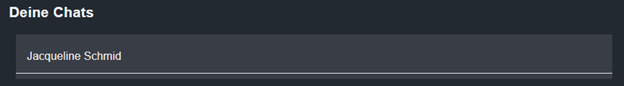


Abbildung 8: Chat-Übersicht. Eigene Darstellung

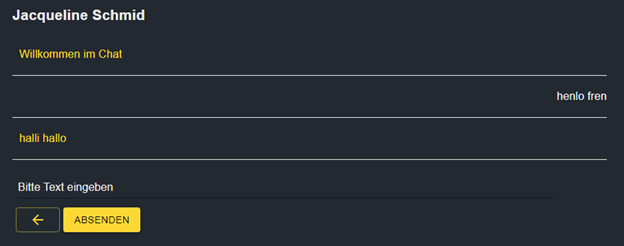


Abbildung 9: Chat. Eigene Darstellung.

Der Button Gruppe erstellen öffnet ein Popup-Menü, in dem der Name und eine kurze Beschreibung der zu erstellenden Gruppe eingegeben werden können.

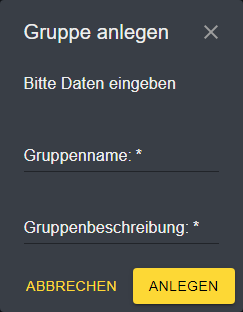


Abbildung 10: Gruppe anlegen. Eigene Darstellung.

Das Matching-Menü führt den Matching-Algorithmus aus und zeigt danach die entsprechenden Vorschläge, geordnet nach Personen und Gruppen an. Diese Liste kann durchgescrollt werden und der Nutzer kann bei jedem Vorschlag eine Kontaktanfrage schicken.

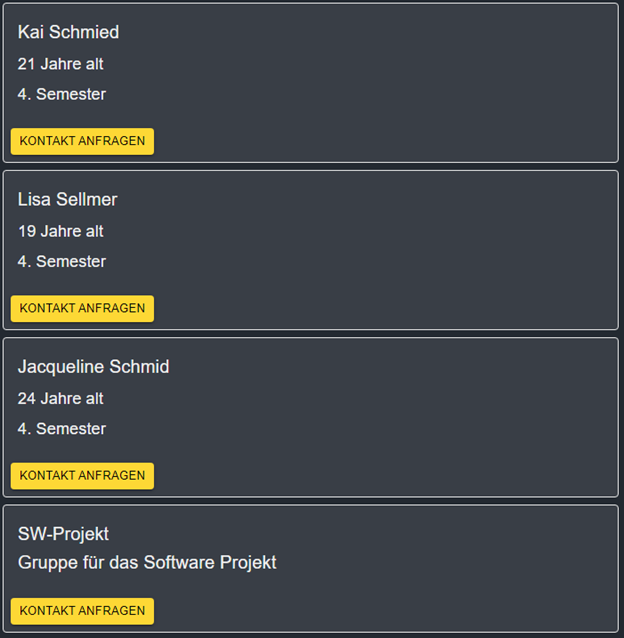


Abbildung 11: Matching-Übersicht. Eigene Darstellung.

Das Anfragen-Menü beinhaltet alle Anfragen, die an den Nutzer oder an die Gruppen des Nutzers geschickt wurden Hier können die Anfragen angenommen oder abgelehnt werden. Werden diese angenommen, können die Nutzer danach miteinander chatten, bzw. der Anfrager wird in die entsprechende Gruppe aufgenommen. Wird die Anfrage abgelehnt, wird einfach die Anfrage gelöscht.

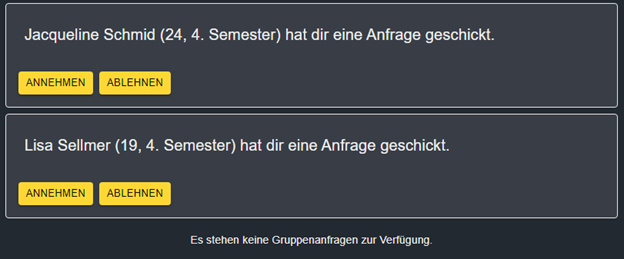


Abbildung 12: Die Anfragenübersicht. Eigene Darstellung.

Im Account-Menü werden die Personen- und Lernprofildaten angezeigt und können hier auch bearbeitet werden. Ebenso gibt es die Möglichkeit, den Account zu löschen.

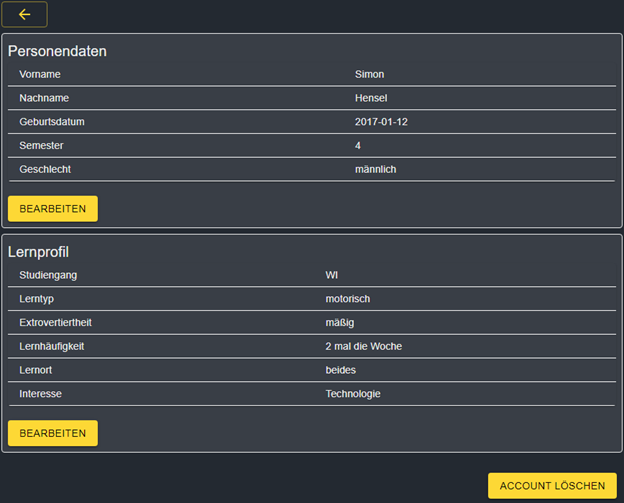


Abbildung 13: Das Account Menü. Eigene Darstellung.

Klick man auf Bearbeiten, öffnet sich ein Popup, über das die jeweiligen Daten bearbeitet werden können.

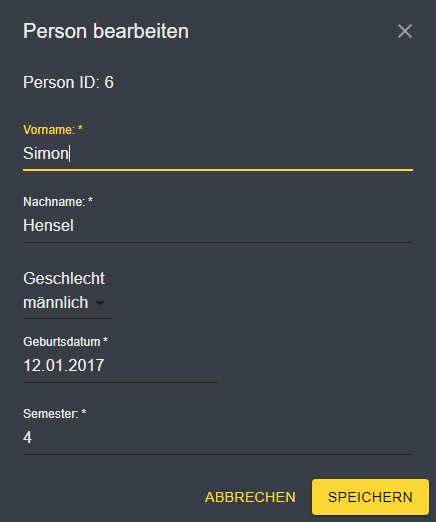


Abbildung 14: Personendaten bearbeiten. Eigene Darstellung.