**Pflichtenheft Prototyp**

Bis Ende März (31.03.17) soll der Prototyp fertig sein, damit das Empfehlungssystem getestet werden kann.

Ausgehend von den qualitativen Interviews und unter Berücksichtigung der Realisierbarkeit soll der Prototyp über folgende drei Funktionen verfügen:

1. Interessen: Der Nutzer soll aufbauend auf von uns erstellten Stichworten seine Interessen angeben können, die dann auf die Vorlesungs-Kategorien abgebildet werden.
2. Zeit: Der Nutzer soll angeben können, in welchem wöchentlichen Zeitraum die Vorschläge stattfinden.
3. Evaluierung: Der Nutzer soll die Möglichkeit haben, die (Vorlesungsevaluationen) Evaluierungsergebnisse einfließen zu lassen. Beispielsweise über den Dozenten.   
   (!!! Nicht machbar !!!)

Das System soll aus mehreren Komponenten bestehen, damit man leicht neue Funktionalitäten in der Zukunft hinzufügen kann. Einige Komponenten, die wir für sinnvoll halten, sind:

1. Display
2. Control
3. Learning
4. Database
5. Feedback
6. Training

Jetzt folgt die Beschreibung der einzelnen Komponenten und deren geplante Funktionalität.

**Display**

Diese Komponente soll sich ausschließlich mit der Darstellung der Inhalte für den Nutzer des Systems beschäftigen. Auf der Homepage des Systems soll es einen Begrüßungsbildschirm geben, wo die Beschreibung des Systems und das Logo des Systems (Modulo) stehen soll. Es soll dann drei weiterführende Links geben. Ein Link soll zur App, ein anderer soll auf eine „Über uns“-Seite und der letzte soll auf eine Seite, wo der verwendete Algorithmus beschrieben wird, führen.

Auf die „Über uns“ Seite sollen allgemeine Informationen über uns als Team zusammen mit ein Foto, über die Jungeakademie und über das Projekt an dem wir arbeiten (in diesem Jahrgang), also was wir mit diesem System machen wollen.

Die Seite, wo der Algorithmus beschrieben wird, soll nur für Interessierte Leute sein und es soll eine Beschreibung oder Dokumentation des Systems und nicht des Projekts sein.

Wenn man auf den Link zur App klickt, soll man sich erstmals mit der TUM Kennung per Shibboleth anmelden. Wichtig wegen Datenschutz ist es, dass dem Nutzer gesagt wird, welche Daten das System braucht und was mit denen gemacht wird. Nachdem der Student sich angemeldet hat, kommt es zur eigentlichen App. Eine genauere Beschreibung der Interface soll nach dem MockUp-Treffen entstehen, hier erstmals eine Möglichkeit.

In der oberen linken Ecke des Bildschirms sollen die Suchfilter für die überfachliche Module sein. Diese sollen sein:

* Ort (Freising, Garching, Hochbrück, Innenstadt u.s.w.)
* Zeitraum (14:00 ‑ 16:00 o.Ä.)
* Anzahl von zu-erwerbenden Credits
* Prüfungstyp und
* Interessen des Studenten (Thema des überfachlichen Moduls)

Alle 5 Filteroptionen sollen als Drop-Down Liste implementiert werden, noch keine Freitextangaben möglich. Nach der Selektion der Filtern sollen diese mithilfe eines Buttons dann dem System übergeben werden.

Der Student soll die Möglichkeit haben, eines oder mehrere Kriterien für das Filtern der vielen Module zu wählen. Falls er gar kein Kriterium wählt, soll das System ihm auf der rechten Seite des Bildschirms alle Module zeigen. Falls er mind. ein Kriterium wählt, dann sollen diese der „Control“ Komponente weitergeleitet werden. Am Ende wird die „Display“ Komponente von der „Learning“ Komponente eine Liste der Module bekommen, die absteigend nach dem Passungsgrad zu den gewählten Kriterien sortiert sind. Aus diesen werden zunächst nur die ersten fünf Module folgenderweise angezeigt.

Auf der rechten Seite des Bildschirms werden die Titel der Module gezeigt und der Student hat die Option auf den Titel zu klicken, um Details über das Modul zu erfahren, wie z.B. Modulbeschreibung, Ort, Zeit, Prüfungstyp, Creditanzahl, (~~Themenkathegorien des Moduls,~~) angestrebte Lernergebnisse, Dozent, Beliebtheit des Moduls, Link zur Anmeldung in TUMonline.

Diese Details werden unterhalb des selektierten Moduls angezeigt und nachdem der Student die Details gelesen hat, gibt es unter diesen Details zwei Buttons: „Interesting“ und „Not for me“. Wird ein Modul geklickt, soll das System sich merken, dass der Student dieses Modul sich angeschaut hat. Es ist wichtig für das Feedback, dass die „Learning“ Komponente bekommt, dass ein Modul zwar angeschaut wurde, aber es weder als „interesting“ noch als „not for me“ vom Student markiert wurde. Die „Feedback“ Komponente soll die Gewichtung des Feedbacks festlegen.

Wenn der Student das Modul als „interesting“ markiert, dann hat er später die Möglichkeit, alle als „interesting“ markierten Module zu sehen und sich in / für einen einzuschreiben / anzumelden. Wichtig ist es, dass nach dem Klicken des Anmeldebuttons der Student eine Fragemeldung bekommt, etwas in der Art: „Wollen Sie wirklich zu dem Modul … anmelden? ‑ Ja, Nein“. Falls der Student ein Modul als „not for me“ markiert, soll das System ihm fragen (um Feedback zu bekommen), warum diese Empfehlung dem Student nicht gefallen hat. Z.B. ob es ein Problem mit dem Thema des Moduls war, oder Ort, Zeit, Credits, Prüfungstyp o.a. Kriterien dem Student nicht gepasst haben. Nachdem der Student das „Feedbackformular“ gefüllt hat, rücken die unteren Empfehlungen einen Platz nach oben (also anstatt des als „not for me“ markierten Moduls) und eine neue Empfehlung kommt, damit man immer 5 Module in dem Block von empfohlenen Modulen hat.

Nachdem der Student einige „interesting“ Module gesammelt hat, soll er die Möglichkeit durch einen Button alle seine selektierten Module zu sehen. Diese Funktionalität soll ähnlich dem Warenkorbprinzip (oder Wunschlisteprinzip) von Amazon sein. Man schaut sich mehrere Artikeln an, selektiert einige aus den empfohlenen Artikeln und am Ende kauft man nur ein Teil von der selektierten Artikeln. Im Fall des Systems wird sich der Student nur zu einem Modul anmelden können. Wichtig ist es, dass nach dem Klicken des Anmeldebuttons der Student eine Fragemeldung bekommt: „Wollen Sie wirklich zu dem Modul … anmelden? ‑ Ja, Nein“. Am Ende des Semesters oder nach der Vorlesungszeit oder nach der Prüfung des Moduls soll das System dem Studenten für Feedback fragen. „Hat Ihnen das Modul gepasst?“, „Hat es deinen Vorstellungen entsprochen?“, „War die Modulbeschreibung passend?“, „Was hat es dir nicht gefallen?“, „Würdest du das Modul an Leuten wie du weiterempfehlen?“ und eine abschließende Markierung „enrolled\_good“ oder „enrolled\_bad“ (wieder wichtig für die „Feedback“ Komponente).

**Control**

Diese Komponente soll sich mit dem Weiterleiten von Daten zu verschiedenen Komponenten beschäftigen. Es soll prinzipiell nur ein kleines Set von Kommunikationswegen geben. Diese sind (ohne Anspruch auf Vollständigkeit…)

* Display -> Learning:  
  Auf diesem Kanal sollen die Suchkriterien, der der Nutzer des Systems gewählt hat, weitergeleitet werden.
* Learning <-> Database:  
  Auf diesem bidirektionaler Kanal sollen die Anfragen an der Modul-Datenbank und die Resultate der Anfragen der Learning Komponente weitergeleitet werden.
* Learning -> Display:  
  Auf diesem Kanal soll die sortierte Liste der empfohlenen Module weitergeleitet werden.
* Display <-> Feedback:  
  Kommunikationaustausch der Markierungen der Module: „interesting“, „not for me“, „seen“, „unseen“, „enrolled\_good“, „enrolled\_bad“
* Feedback <-> Learning:  
  Auf diesem Kanal soll das Feedback zur gegebenen Empfehlung (Sortierung der Module) der „Learning“ Komponente weitergeleitet werden.
* Training <-> Learning:

Auf diesem bidirektionaler Kanal soll den Kommunikationaustausch für das Trainieren das Systems stattfinden.

* Training <-> Database:  
  Auf diesem bidirektionaler Kanal sollen die Anfragen an der Personen-Datenbank und die Resultate der Anfragen der Training Komponente weitergeleitet werden.

**Learning**

Diese Komponente soll sich mit dem Sortieren der Module basierend auf den vom Nutzer angegebenen Kriterien beschäftigen. Diese Komponente kann sowohl ein „Reinforcement learning“ als auch ein „Supervised learning“ Ansatz für das Lernen verwenden. Prinzipiell werden drei Lernprozesse stattfinden:

1. Das Lernen des Mappings „Interessen des Studenten“ -> „Themenkategorien der Module“
2. Lernen, wie die Kriterien eigentlich interpretiert werden müssen. Man kann etwa Variationen zu den Eingaben einführen (da der Student vielleicht doch ein interessantes Modul nehmen will, auch wenn es vllt. weniger Credits, als in den Kriterien angegeben, hat). So kann man z.B. wenn der Nutzer Creditanzahl als 5 angegeben hat, das als das Intervall [4, 6] interpretieren und so die Suchkriterien anpassen, so dass bessere Vorschläge kommen.
3. Das Lernen, wie man die Module nach den angegebenen Kriterien sortiert (Sind Credits wichtiger als Ort? Ist das Thema des Moduls am wichtigsten? u.s.w.)

Diese Komponente soll mit der Modul-Datenbank interagieren, um die Liste der Module zu bekommen und sie anschließend nach den angegebenen Kriterien sortieren und der „Display“ Komponente zum Anzeigen weiterleiten.

**Database**

Diese Komponente soll sich mit dem Verwalten der voraussichtlich zwei gebrauchten Datenbanken beschäftigen. Eine Datenbank soll alle Informationen über die Module verwalten und beinhalten (Modul-Datenbank) und die andere soll Testpersonen für das Trainieren der „Learning“ Komponente beinhalten (Personen-Datenbank).

Die Modul-Datenbank soll die Module beinhalten, die Kategorien der Module und die Liste der Interessen, die der Nutzer als Filterkriterium der Module angeben kann.

* Module: id (PK), title, description, exam, credits, location, time, lecturer, category1 (FK), category2 (FK), …, category5 (FK)
* Category: id (PK), category\_name,
* Interests: id (PK), interest\_name

Mögliches Problem: Module <-> Category ist eine N:M Beziehung. Wie lösen wir das?

Die Personen-Datenbank soll viele Personen mit verschiedenen Wünsche, Persönlichkeiten und Präferenzen (thematisch, zeitlich, örtlich u.v.m.) und die „richtigen Ergebnisse“ beinhalten. Die „richtigen Ergebnisse“ sind die akzeptablen Empfehlungen (Themenbereiche) von Modulen (also die Module, die diese Person (Persönlichkeitstyp) annehmen wurde) und das „Akzeptierungsintervall“ für andere Kriterien (z.B. bei angegebenen 5 Credits, kann ein Modul, das 3 bis 6 Credits hat angenommen werden). Ein Ausschnitt von möglichen Persönlichkeitstypen kann sein:

* Strict
* Loose
* Curious
* Credit-oriented
* Lazy
* Nerd 
* Do-it-all
* Time-pressured
* Detail-oriented
* Anything works
* Easy exam
* Only written exam
* The harder, the better
* And many more

**Feedback**

Diese Komponente soll sich mit der Interpretation der Aktionen des Nutzers beschäftigen, nachdem er die Empfehlungen von Modulen bekommt. Er soll eine Liste von folgenderweise eingeteilten Modulen verwalten und gewichten (interpretieren), wie gut die Empfehlung zu dem Nutzer gepasst hat (damit die „Learning“ Komponente daraus lernen kann).

* Interesting\_courses
* Not\_for\_me\_courses
* Seen\_courses
* Not\_seen\_courses
* Selected\_course\_good
* Selected\_course\_bad

Diese Listen bestimmen den positiven oder negativen Einfluss auf die Empfehlung, möglicherweise nach folgender Schema:

1. Positiver Einfluss:
   1. Selected\_course\_good
   2. Interesting\_courses
2. Kein Einfluss:
   1. Not\_seen\_courses
3. Negativer Einfluss:
   1. Not\_for\_me\_courses
   2. Selected\_course\_bad
   3. Seen\_courses

Diese Komponente soll auch die Angaben des Nutzers vom bekommenen Feedback (z.B. bei „not\_for\_me\_courses“ die Gründe, warum die Empfehlung nicht gepasst hat) interpretieren.

Das ganze Feedback wird dann zusammengefasst und der „Learning“ Komponente weitergeleitet, um die Empfehlungen für künftige Nutzer zu verbessern.

**Training**

Diese Komponente soll sich mit dem Trainieren der „Learning“ Komponente beschäftigen. Dies bedeutet, dass die Zeit, die nach Anmeldung eines Studenten zu einem Modul bis die Feedbackrunde des gewählten Moduls reduziert wird, damit man mehr Zeit zum trainieren hat. Diese Komponente soll mit der Personen-Datenbank interagieren und den Systemablauf simulieren, als ob die gewählte Person aus der Datenbank das System verwenden würde. Um das Trainingprozess zu starten, braucht man ADMIN Rechte auf das System (Anmeldung).