**Mögliche Issues für unsere Meilensteine**

Planungsdokumente fertigstellen:

* Festlegen auf ein grafisches Tool (falls Gantt-Diagramm genutzt werden soll o. ä.)
* Oder gleich ein Projektmanagement-Tool, dass zusätzlich für die Planung genutzt werden sollte neben GitHub
* Grafische Darstellung von Metriken, Analysen und geschätzte Zeiten für die einzelnen Tasks
* Herausarbeiten der Tasks und Sub-Tasks
* Feingranulare Zeiteinschätzungen
* Risikofaktoren erörtern
* Wie soll das Risk Management durchgeführt werden (Tools usw.)
* Kommunikationsmittel (als Slack festgelegt)
* Sitzungen (bereits getan)
* Designspezifikationen?
* Integration von Features nach GitFlow Richtlinien
* Sprache festlegen (Java?)
* Build-Tool (Maven?)
* Werden wir Verschlüsselungsmethoden einrichten?
* Containerisieren wir? Wenn ja, welches Tool?
* Automatisierungstools festlegen?
* Framework festlegen (Hibernate?)
* Qualitätssicherung per Pull-Request
* Allokation der jeweiligen Tasks auf die Teammitglieder
* Upload aller erstellten Planungsdokumente
* Workload pro Teammitglied ermitteln?
* Rotation der Zuständigkeiten vorab planen?
* DBMS?

Anforderungen definieren:

* Anforderungen aus der Aufgabenstellungen
* Definieren der Stakeholder
* Was waren die Probleme des alten Systems
* Zentrale Nutzen des Systems?
* Nur eine bestimmte Anzahl von Studenten zulassen
* Zulassungsbescheide werden automatisiert versendet nach Entscheidung des Zulassungsausschusses
* Unterstützer können Empfehlungsschreiben hochladen
* Bewerber können relevante Daten übermitteln
* Zulassungsausschuss benötigt eine View über eingespeiste Daten
* Übersichtliche benutzerfreundliche View
* Client-Server Modell
* Wie stellen wir den Server zur Verfügung (oder der Einfachheit halber nur lokal)
* Client hat Zugriff über Browser (Web-Server wird benötigt)
* Verschlüsselungstechniken?
* Hardwareanforderungen ermitteln?
* Datenbankanbindung

SW Architektur definieren:

* Programmiersprache festlegen
* Framework festlegen
* Build-Tool festlegen
* Web-Server festlegen
* Installierbar oder nur binarys zur Verfügung stellen
* Architektur grafisch festhalten
* Ggfl. Verschlüsselungsmethode
* DBMS festlegen
* Ggfs. Plugins für Schnittstellen bestimmen
* Übermittelte Daten auf DB speichern
* Via Framework eine View bereitstellen, um Daten der Studieninteressierte übersichtlich zu gestalten
* OS für Server festlegen, falls nicht Cloud Lösung oder On-Premise genutzt wird
* Upload von pdf-Dateien ermöglichen
* Einschränkungen für uploadbare Dateien definieren
* Können Dateien wieder entfern werden nach dem Upload, um noch mal neu hochzuladen?
* Falls wir Tools aus CI/CD nutzen sollten, welche werden das sein?
* Wird der Student sich einloggen müssen für eine Bewerbung?
* Wenn ja, dann müssen Accounts der Studieninteressierte verwaltet werden

System entwickeln:

* Einarbeiten in die Softwaretools (besser gesagt lbd)
* Die jeweiligen Komponenten entwickeln
  + Issues zu den Arbeitsschritten, die zur Fertigstellung beitragen erstellen
* Schnittstellen der Komponenten bilden
* Hardware bzw. Cloud -Service o.ä. konfigurieren
* Installationsfähiges Softwareartefakt erstellen
* Deployen der Anwendung
* Web-Server konfigurieren
* DBMS installieren und konfigurieren
* Ggfl. SSH oder ähnliches einrichten
* Hotfixes

Testing:

* Testen der Software
* Bugfixes
* Qualitätssicherung einzelner Komponenten
* Stabilität des gesamten Systems überprüfen
* Stresstest? (auch wenn die Kapazitäten nicht wirklich stark belastet werden können im Rahmen des Kurses)

Bereitstellung Repository (Abgabevorbereitung):

* Den Anforderungen entsprechend die GitHub Repo, Doku, etc. vorbereiten
* Alle überprüfen und segnen ab