Projektantrag SEPM QSE5

Dionysus Libation Lab

Entwicklerteam

Matrikelnummer	**Name**	**Rolle**	**Stellvertretende Rolle**
12119840	Elias Eckermann	Testleiter	Technischer Architekt
12024722	Jonas Epner	UI / UX	Dokumentationsmanager
11921623	Valentin Gobl	Technischer Architekt	Team Koordinator
12024734	Marvin Kleinlehner	Team Koordinator	UI / UX
01549616	Katarina Pjanic	Dokumentationsmanager	Testleiter

Ausgangssituation

Studenten sind bemüht, mit begrenzten Ressourcen das meiste aus ihrer Situation zu machen. In der Zubereitung von Cocktails gibt es eine Vielzahl von Zutaten, viele Cocktails haben hierbei starke Überschneidungen, aber gänzlich andere Geschmacksprofile. Es kann oft für einen Gastgeber schwierig sein, nachhaltig zu arbeiten und trotzdem ein vielseitiges Sortiment anzubieten.

Projektbeschreibung

Wir möchten eine Webanwendung entwickeln, die Gastgeber:innen die Möglichkeit bietet, eine Liste der Zutaten einzugeben, die sie in ihrem Vorrat haben. Die Anwendung zeigt dann an, welche Cocktails mit den verfügbaren Zutaten zubereitet werden können. Darüber hinaus können Gastgeber:innen Empfehlungen für den Einkauf erhalten, die darauf abzielen, die bereits vorhandene Palette an Cocktails zu erweitern, indem möglichst viele verschiedene Cocktails mit den verfügbaren Zutaten angeboten werden.

Es wird auch eine weitreichende Suchansicht angeboten, mit der sich die Benutzer:innen durch die Welt der Cocktails suchen können. Sie können nach verschiedenen Tags filtern und auf diese Weise neue Lieblinge entdecken.

Die Benutzer:innen können sich in Gruppen zusammenschließen, bei denen der Gastgeber oder die Gastgeberin in jeder Runde wechselt. Jeder Gastgeber oder jede Gastgeberin wählt für das Treffen eine Auswahl an Cocktails aus, die angeboten werden. Zu der manuellen Auswahl kann auch noch ein Algorithmus hinzugezogen werden, der die Präferenzen der Gäste mit einbezieht. Die Gäste können die Cocktails bewerten, die sie probiert haben, indem sie Likes oder Dislikes vergeben. Diese Bewertungen dienen dazu, dem Gastgeber oder der Gastgeberin Feedback zu geben und den Gästen dabei zu helfen, eine Übersicht über die Cocktails zu behalten, die sie mögen.

Unser Ziel ist es, gesellige Zusammenkünfte zu fördern und den Usern eine breite Palette an Cocktail-Erlebnissen zu bieten, indem wir die vorhandene Palette erweitern.

Zielgruppe

Unsere Webanwendung richtet sich an eine vielseitige Zielgruppe, die ein gemeinsames Interesse an Cocktails teilt. Sie zielt auf Menschen ab, die Freude am Mixen und Entdecken von neuen Cocktails haben. Zusätzlich möchten wir Menschen ansprechen, die gerne unterhaltsame Abende mit Freunden verbringen, während sie köstliche Cocktails genießen. Außerdem ist unsere Anwendung auch für Hobby-Barkeeper beziehungsweise Gastgeber von Cocktail Partys gedacht, die nach Inspiration für neue Cocktailkreationen suchen oder herausfinden möchten, welche Cocktails sie mit den aktuell verfügbaren Zutaten zubereiten können.

Userstories

Die Kunden-Priorität ist spezifiziert als "N" für niedrig, "M" für mittel und "H" für hoch. Der Aufwand ist, im Hinblick auf Agile Methoden (Iceberglist), als abstrakte "Story Points" auf einer Skala von 1 bis 9 geschätzt.

Id	Feature	Beschreibung	Kundenpriorität	Aufwand
1	Registrierung der User	User können sich mit E-Mail und Usernamen registrieren. Diese Daten werden mit berücksichtigung auf Security in der Datenbank gespeichert	H	3
2	Gruppen erstellen	User können Gruppen erstellen, dadurch wird der Ersteller zum Host. Andere User können in die erstellte Gruppe eingeladen werden. Dies ist nur als Host möglich.	H	4
3	Gastgeber	Der derzeitige Gastgeber kann	M	2

	bestimmen	einen neuen Gastgeber bestimmen. Dabei verliert er jedoch den Gastgeber-Status.		
4	Cocktails durchsuchen	User sollen alle Cocktails unabhängig von ihrer Verfügbarkeit durchsuchen können. Es soll auch möglich sein, Tags in dieser Suche zu verwenden.	H	4
5	Präferenzen setzen	Gäste können Präferenzen in Form von Tags angeben (z.B. Süß). Diese werden dem Gastgeber dann bei der Zusammenstellung des Angebots angezeigt, um eine gute Abstimmung auf die Gruppe zu ermöglichen.	M	3
6	Zutaten angeben	Bevor der Gastgeber eine Cocktail-Empfehlung vom System erhält, muss er seine bereits vorhandenen Zutaten angeben. Gäste können auch angeben, dass sie eine bestimmte Zutat mitbringen werden. Zutaten werden Tags anhand von ihrem Geschmacksprofil zugewiesen.	H	5
7	Zubereitbare Cocktails anzeigen	Der Gastgeber soll, basierend auf der angegebenen Zutatenliste, alle Cocktails die zubereitbar sind angezeigt bekommen.	H	7
8	Einkaufsempfehlung	Ein Gastgeber soll basierend auf seinen verfügbaren Zutaten einen Vorschlag bekommen, mit welchem er seine zubereitbaren Cocktails maximieren kann. (Maximierungsproblem)	H	9
9	Cocktailkarte erstellen	Der Gastgeber kann eine Cocktailkarte für die Gruppe erstellen. Auf dieser Karte sollen nur Cocktails aufgeführt sein, die der Gastgeber zubereiten kann.	H	3
10	Cocktails bewerten	Gäste können Cocktails, die angeboten wurden, mit einem	N	1

		Like oder Dislike bewerten. Das dient dem Gastgeber als Feedback		
(11)	AI generierte Cocktails mit einbeziehen	Bei den Vorschlägen soll es möglich sein, anzugeben, ob AI generierte Cocktails in den Vorschlag mit einbezogen werden sollen (erfundene Cocktails).	N	5

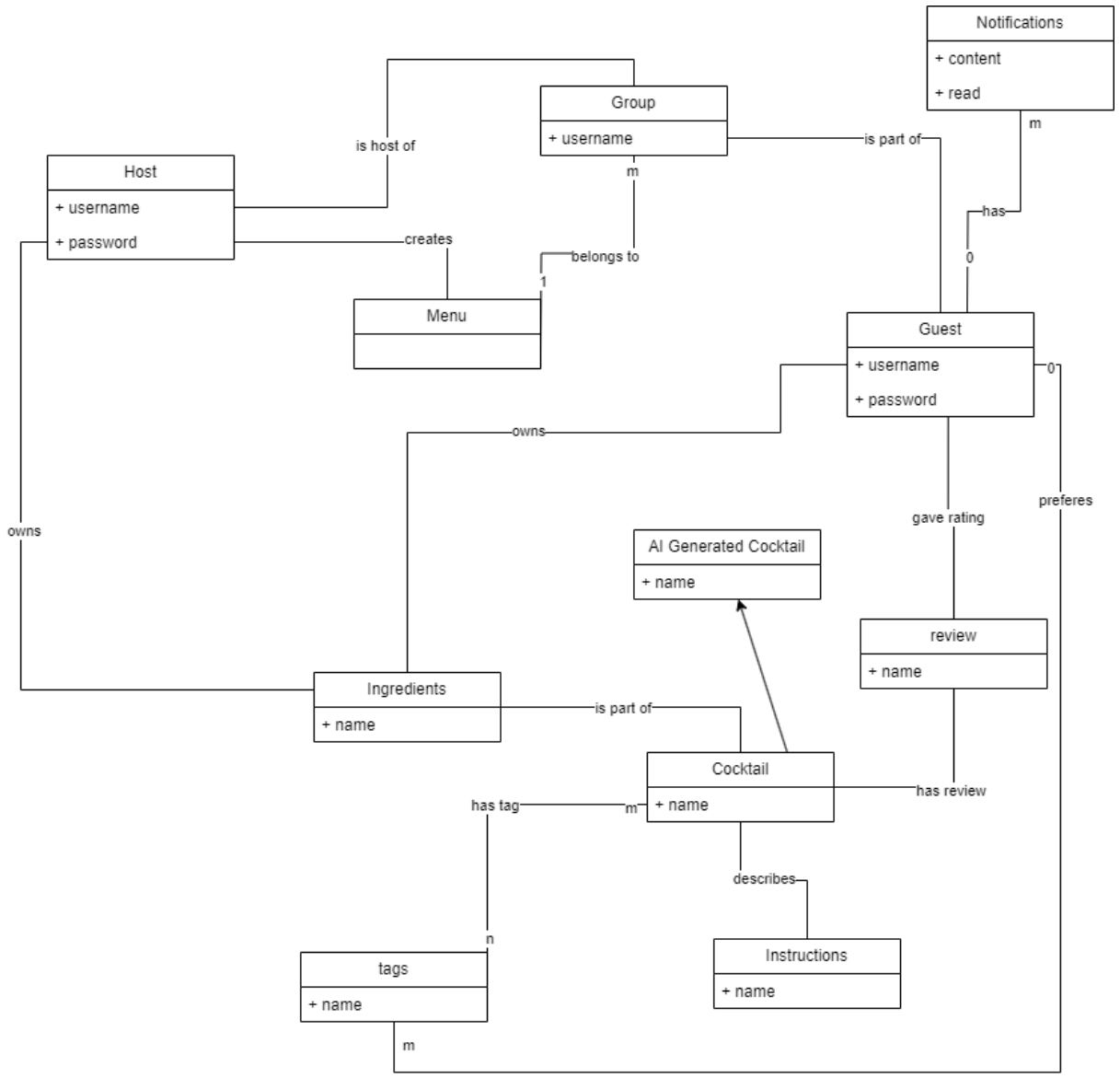
Komplexe Komponente

Die Vorschläge für die Cocktailkarten sollen nach folgenden Parametern spezifiziert werden.
Parameter:

- max Anzahl an neuen Zutaten welche der Gastgeber bereit ist zu kaufen (optional)
- genaue Anzahl an verschiedenen Cocktails auf der Karte (optional)
- (US10 optionales feature AI generierte cocktails aktivieren)
- anzahl der verschiedenen Vorschläge (max 5)
- Präferenzen der anderen Gruppenmitglieder sollen berücksichtigt/gewichtet werden

Mithilfe eines Algorithmus sollen dem User optimale Vorschläge gegeben werden.
Des Weiteren sollte darauf geachtet werden, dass die verschiedenen Vorschläge nicht immer dieselben Cocktails enthalten.
=> Greedy vs Dynamic Programming
Combinatorial optimization problem (NP-schwer)

Domänenmodell



Abgrenzungen (Nicht-Ziele)

Bestellsystem

Das Projekt hat zum Ziel, Cocktails basierend auf vorhandenen (und möglichst wenigen neu gekauften) Zutaten vorzuschlagen. Es soll hier kein Einkaufs- bzw. Bestellsystem für Zutaten implementiert werden. Der Fokus liegt auf der Verwendung von vorhandenen Zutaten.

Lagerorganisation

Es soll lediglich mit Zutaten für Cocktails gearbeitet werden. Es soll allerdings keine Lagerverwaltung für alle möglichen Lebensmittel oder sogar andere Produkte implementiert werden.

Event-Management

Das Projekt soll die Bildung von Gruppen ermöglichen, sowie das Bestimmen eines Gastgebers. Das Managen einer Veranstaltung oder die Verwaltung von Veranstaltungsorten, etc. ist nicht Ziel des Projekts.

Aufwendige Benutzerverwaltung

Die Benutzerkonten werden gebraucht, um private Gruppen bilden zu können und in diesen Gastgeber zu bestimmen. Eine Umfassendere Benutzerverwaltung mit verschiedenen Rollen, ist nicht Ziel des Projekts.

Mobile App

Das Projekt soll eine Web-Anwendung werden, die besonders auf Computern und Tablets genutzt wird. Die Seite soll auch auf Smartphones gut funktionieren, aber nicht speziell für diese optimiert sein.

Soziales Netzwerk

Das Projekt soll kein Soziales Netzwerk werden und daher keine Funktionen wie aufwendige Benutzerprofile (die über die Cocktail interessen hinaus gehen), private Nachrichten, Freundschaftsanfragen oder ähnliches unterstützen.

Planner für Cocktailbars

Das Projekt soll keine spezielle Funktionalität für Cocktail Bars, Clubs oder Restaurants haben, sondern sich auf kleine private Runden fokussieren.

Tutorials

Das Projekt soll keine Tutorial-Seite mit Live-Streams, Videos oder Erklärungen zum machen von Cocktails sein, sondern sich auf das Vorstellen von Cocktails und probieren dieser konzentrieren.

Ranglisten

Es soll kein Wettbewerb artiges Cocktail machen gefördert werden, sondern ein gemeinsames Entdecken von neuen Rezepten. Insofern soll man zwar Cocktails bewerten können, um sich zu späteren Zeitpunkten an besonders gute Cocktails erinnern zu können, bzw. diese wiederzufinden, und auch empfehlen zu können. Es soll allerdings keine Ranglisten der besten Cocktails oder ähnliches geben.

Live-Verfügbarkeit und Dritte

Die Plattform soll lediglich anzeigen, welche Cocktails mit welchen Zutaten möglich sind, soll allerdings keine Verfügbarkeits-Informationen aus Geschäften verwenden oder Drittanbieter einbinden, um die Preise für Cocktails zu vergleichen oder gesponsorte Geschäfte zu empfehlen.

Funktionalen Anforderungen

Anwendungsfall Übersicht:

1. *Registrierung* (**H**): Unbedingt erforderlich, kritisch
2. *Gruppenbildung* (**H**): Unbedingt erforderlich, kritisch
3. *Gastgeberfunktion* (**M**): nur Grundfunktionalität notwendig
4. *Cocktail-Suche* (**H**): Unbedingt erforderlich, kritisch
5. *Präferenzangaben* (**M**): nur Grundfunktionalität notwendig
6. *Zutatenverwaltung* (**H**): Unbedingt erforderlich, kritisch
7. *Zubereitbare Cocktails* (**H**): Unbedingt erforderlich, kritisch
8. *Einkaufsempfehlung* (**H**): Unbedingt erforderlich, kritisch
9. *Cocktailkarte-Erstellung* (**H**): Unbedingt erforderlich, kritisch
10. *(Cocktails-)Bewertung* (**N**): Erweiternde Features, können auch nach dem Projekt implementiert werden
11. *AI generierte Cocktails* (**N**): Erweiternde Features, können auch nach dem Projekt implementiert werden

Iceberglist:

Id	Feature, Aktor	Anwendungsfälle	Kunden-Priorität	Auf-wand	Ver-sion	Zustän-diger
1.1	Registrierung, Anonym	Account anlegen	H	3	1	Kleinlehner
1.2	Registrierung, User	Neues Passwort anfordern	H	2	1	Epner
1.3	Registrierung, User	Account löschen	H	2	1	Epner
2.1	Gruppenbildung, User(Gastgeber)	Gruppe erstellen	H	4	1	Eckermann
2.2	Gruppenbildung, Gastgeber	User zu Gruppe einladen	H	3	1	Gobl
2.3	Gruppenbildung, User(Gast)	Gruppe beitreten (nur mit Einladung)	H	3	1	Gobl
2.4	Gruppenbildung, Gast	Gruppe verlassen	M	2	1	Eckermann
2.5	Gruppenbildung, Gastgeber	Gast suchen	M	4	1	Pjanic
2.6	Gruppenbildung, Gastgeber	Gast aus der Gruppe entfernen	M	2	1	Pjanic
2.7	Gruppenbildung, Gastgeber	Gruppe löschen	H	2	1	Pjanic

2.8	Gruppenbildung , Gastgeber, Gast	Gruppe ansehen	M	3	1	Kleinlehner
3.1	Gastgeberfunktion, Gastgeber	Neuen Gastgeber bestimmen (mit Benachrichtigung)	M	2	1	Eckermann
4.1.1	Cocktail-Suche, User	Cocktails durchsuchen	H	4	2	Epner
4.1.2	Cocktail-Suche, User	Tag in der Cocktails-Suche wählen	M	3	3	Epner
5.1	Präferenzangaben, Gast	Tag suchen	M	2	3	Kleinlehner
5.2	Präferenzangaben, Gast	Tag als Präferenz setzen	M	3	3	Kleinlehner
5.3	Präferenzangaben, Gast	Tag als Präferenz entfernen	M	2	3	Gobl
5.4	Präferenzangaben, Gastgeber	Präferenzen ansehen	M	3	3	Gobl
6.1	Zutatenverwaltung, Gastgeber, Gast	Zutat suchen	M	2	2	Pjanic
6.1.2	Zutatenverwaltung, Gastgeber	Vorhandene Zutat (für Cocktail-Empfehlungen) angeben	H	5	2	Pjanic
6.1.3	Zutatenverwaltung, Gast	Bestimmte Zutat mitbringen	H	5	2	Eckermann
6.2	Zutatenverwaltung, Gastgeber	Vorhandene Zutaten ansehen	H	3	2	Eckermann
7.1	Zubereitbare Cocktails, Gastgeber	Zutaten-kompatible Cocktails anzeigen	H	5	2	Kleinlehner
7.2	Zubereitbare Cocktails, Gastgeber	Cocktail (aus der Zutaten-kompatible Coctails-Liste) ansehen	M	3	2	Kleinlehner
8.1	Einkaufsempfehlung, Gastgeber	Zutatenbasierte Maximierungsvorschlag der Cocktail-Zubereitung ansehen	H	9	4	Kleinlehner
8.2	Einkaufsempfehlung, Gastgeber	Vorschläge für Lebensmitteleinkauf anzeigen	H	5	4	Epner

9.1	Cocktailkarte-Erstellung Gastgeber	Cocktailkarte erstellen	H	3	3	Pjanic
9.2	Cocktailkarte-Erstellung Gastgeber	Zubereitbare Cocktails suchen	H	3	3	Pjanic
9.3	Cocktailkarte-Erstellung Gastgeber	Cocktail zur Cocktailkarte hinzufügen	H	3	3	Eckermann
9.4	Cocktailkarte-Erstellung Gastgeber	Cocktail aus der Cocktailkarte entfernen	H	3	3	Epner
9.5	Cocktailkarte-Erstellung Gastgeber, Gast	Cocktailkarte ansehen	M	2	3	Gobl
10.1	Bewertung, Gast	Angebotene Cocktails (mit einem Like oder Dislike) bewerten	N	2	4	Eckermann
10.2	Bewertung, Gastgeber, Gast	Angebotene Cocktails (mit Bewertungen) ansehen	N	2	4	Pjanic
10.3	Bewertung, Gastgeber	Angebotene Cocktails (nach Bewertungen) sortieren	N	4	4	Epner
11.1	AI generierte Cocktails, Gastgeber	mit AI einen Cocktail erstellen	N	5	4	Gobl

Nicht funktionalen Anforderungen

Leistung und Verfügbarkeit

Die Anwendung soll performant sein: Der Nutzer sollte bei einfachen Anfragen nie länger als 2 Sekunden warten. Bei längeren Berechnungen auf der Serverseite muss ein Lade-Indikator angezeigt werden.
Die Anwendung muss stabil sein und darf nicht einfach abstürzen.

Wartbarkeit und Dokumentation

Die Anwendung soll zu einem späteren Zeitpunkt möglichst gut wartbar sein. Damit dies möglich ist, muss die Architektur genau eingehalten werden. Außerdem muss das Programm gut dokumentiert sein: Für alle Schnittstellen muss JavaDoc geschrieben werden.

Datensicherheit

Informationen innerhalb einer Gruppe dürfen nur für Mitglieder dieser Gruppe sichtbar sein, sowie private Informationen eines Nutzers (insbesondere Anmeldedaten) auch nur für diesen sichtbar sein dürfen.

Sicherheit

Passwörter müssen mit einer sicheren Hashfunktion gehashed werden. Es dürfen auf keinen Fall Passwörter im Klartext gespeichert oder übertragen werden. Nutzer müssen in der Lage sein, Passwörter zu ändern und zurückzusetzen. Die Authentifizierung muss sicher sein und Kommunikation zwischen Nutzer und Server verschlüsselt.
Alle Eingaben von Nutzern müssen validiert werden. Es dürfen keine Fehlerhaften Daten gespeichert werden.
Zugriffe auf die Datenbank müssen sauber sein. Es darf keine Möglichkeit einer SQL-Injection geben.

Benutzeroberfläche

Die UI muss übersichtlich gestaltet sein. Die wichtigsten Features müssen deutlich sichtbar sein. Sie sollten auf der jeweiligen Seite in maximal 2 Klicks erreichbar sein.
Es darf nicht zu viel Funktionalität auf wenig Platz zusammengefasst werden.
Die UI muss intuitiv sein (Elemente müssen den Nutzern erzählen, wie sie bedient werden wollen: Slider gezogen, Buttons geklickt,...)
Die Suche nach Cocktails und Benutzern muss sich wie im Punkt Leistung und Verfügbarkeit definiert verhalten.

Kompatibilität

Die Anwendung muss für möglichst viele Nutzer bedienbar sein. So müssen zumindest die Browser Chrome (ab Version 117) und Firefox (ab Version 117) unterstützt werden.

Qualität

Das Projekt muss mit hoher Qualität umgesetzt werden. So muss das Programm ausreichend getestet werden und es müssen Logfiles geschrieben werden, um Fehler schnell zu finden. Die Programmarchitektur muss strikt eingehalten werden. Die Fehlerbehandlung muss durchdacht sein. Es wird ein statischer Code-Checker verwendet, um eine gewisse Lesbarkeit und Qualität des Codes sicherzustellen.

Lieferumfang und Abnahme

Während der Projektlaufzeit und vor allem beim Abschluss des Projektes wird dem Kunden die erstellte Software sowie dazugehörige Dokumentation übergeben. In diesem Abschnitt werden die Softwarekomponenten und Dokumentation beschrieben und eine Abgrenzung des Lieferumfangs durchgeführt.

Software

Nach Projektabschluss werden dem Kunden folgende Kernkomponenten der Software in Produktionsfähigen Zustand übergeben:

- Angular Webapplikation Build File
- Java Backend Jar File

Artefakte & Projektdokumentation

- Domänenmodell & Komponentendiagramm
- Funktionale Anforderungen
    - Anwendungsfallbeschreibung
    - Anwendungsfalldiagramme, Pakete & Aktorenhierachie
- Benutzerhandbuch
- Datenbankbeschreibung & ER Diagramm
- Mockups zur Darstellung der Webansicht
- Präsentation zu den Reviews (MRs)

Meilensteine

Meilenstein 2: Erster Systemtest (Hello World)

- Artefakte des Laufenden Projektmanagement (Besprechungsprotokolle, Stundenliste, Iceberglist, Projektstrukturplan (Work Breakdown Structure auf Arbeitspaketebene))
- Funktionale Anforderungen (Übersicht über alle Anwendungsfälle, Beschreibung der Anwendungsfälle, Anwendungsfalldiagramme, AkteurInnenliste, AkteurInnenhierarchie)
- Komponentendiagramm
- UI-Skizze und Beschreibung (Figma)
- “Hello World” Protoyp mit GUI: erster Systemtest
- Testartefakte (Testplan)
- Datenbankbeschreibung (ER-Diagramm, Beschreibung der Attribute)

Meilenstein 3: GUI Prototyp mit DAO Tests, 60% der User Stories implementiert

- Artefakte des Laufenden Projektmanagement
- Anwendungsfallbeschreibung überarbeitet und abgeschlossen
- Datenbankbeschreibung abgeschlossen
- “Alpha Version”: Prototyp mit DAO Tests und 60% der Funktionalen Anforderungen (User Stories) implementiert
- UI Skizzen/Design abgeschlossen
- Komponentendiagramm überarbeitet und abgeschlossen

Meilenstein 4: 80% der User Stories implementiert, Service-Schicht Tests

- Artefakte des Laufenden Projektmanagement
- “Beta Version”: 80% der User Stories implementiert mit Service-Schicht Tests

Meilenstein 5: 100% der User Stories implementiert, Akzeptanztests

- Artefakte des Laufenden Projektmanagement
- “General Availability Version”: funktionierendes Produkt

Arbeitsstruktur und Rollenverteilung

Das Projektteam besteht aus 5 Entwicklern, die Expertenrollen (und damit verbundene Horizontale Verantwortlichkeiten) wurden bereits im Projektvorschlag definiert. Für dieses Projekt wird eine angepasste Variante von SCRUM verwendet.

Rollenverteilung

Aufgrund dieser Expertenrollen muss die Projekt- (SCM, Tracker) und Produktinfrastruktur (Maven) aufgebaut werden. Auch das verfassen von Artefakten bzw. Dokumentation unterliegt bestimmten Experten (siehe Grober Projektplan), jedoch reviewen und beitragen muss jeder.

Horizontale Verantwortlichkeiten

Team Koordinator

Team Koordinatoren müssen gute Projektmanagement Kenntnisse haben und sind für die Aktualität der Artefakte des laufenden Projektmanagement (z.B. Projektplan) verantwortlich.

Teamkommunikation: Der Teamkoordinator ist verantwortlich für die Förderung einer klaren und effektiven Kommunikation innerhalb des Teams. Dies beinhaltet die Organisation von Teammeetings, das Teilen von Informationen und das Sicherstellen, dass alle Teammitglieder auf dem gleichen Stand sind.

Arbeitsplanung und -koordination: Der Teamkoordinator koordiniert die Arbeit zwischen den Teammitgliedern, um sicherzustellen, dass alle Ziele im Zeitplan erreicht werden. Weiters bestimmt er die Arbeitsverteilung.

Controlling und tracking: Der Teamkoordinator ist verantwortlich, dass von jedem Teammitglied eine Stunde-Liste geführt wird. Weiters ist er für die Arbeitsverteilung zuständig und muss kontrollieren, dass diese eingehalten wird.

Ansprechpartner: Der Teamkoordinator ist für den Statusbericht verantwortlich und dient als zentrale Ansprechperson für den Kunden.

Testleiter

Testinfrastruktur: Die Testinfrastruktur bildet das Fundament für eine effektive Qualitätssicherung. Hierzu gehören Testbibliotheken sowie Testdaten in einer separaten Datenbank. Die Integration mit dem Spring Framework ermöglicht ein effizientes Management der Testdaten. Ein zentrales Anliegen ist die strikte Trennung des Testcodes vom Produktionscode, um die Integrität der Tests sicherzustellen.

Überwachung von Integrations- und Systemtests: Die kontinuierliche Zusammenarbeit mit dem Technischen Architekten ist entscheidend, um eine effektive Überwachung von

Integrations- und Systemtests zu gewährleisten. Diese enge Abstimmung ermöglicht eine zeitnahe Identifizierung und Lösung potenzieller Probleme.

Regelmäßige Überprüfung aller Unit-Tests: Die regelmäßige Überprüfung aller Unit-Tests stellt sicher, dass auf der kleinsten Entwicklungsstufe eine hohe Codequalität gewährleistet ist. Dies umfasst die Validierung der Testergebnisse und die Anpassung der Tests an Änderungen im Produktionscode.

Dokumentationsmanager

Time Tracking: Der Dokumentationsbeauftragte ist verantwortlich, dass das Time Tracking übersichtlich und nachvollziehbar gestaltet ist. Dies beinhaltet die Kontrolle von Eintragungen und das Überprüfen von Time Tracking Issues.

Dokumentation von Meetings: Das Mitschreiben und Zusammenfassen von Meetings und die darin besprochenen Thematiken müssen vom Dokumentationsbeauftragten sichergestellt werden. Dadurch wird sichergestellt, dass Komponenten wie besprochen implementiert werden und etwaige Unklarheiten geklärt werden können.

Dokumentation von Projektstand: Das Projekt sollte mitdokumentiert werden, um den Kunden auf Anfrage einen derzeitigen Stand präsentieren zu können. Dies sollte den aktuellen Implementierungsstand enthalten und bereits implementierte Funktionalität aufweisen.

UI / UX

Der UI/UX-Verantwortliche ist für die kritische Schnittstelle zwischen Benutzer und Software verantwortlich. Diese Schlüsselrolle erstreckt sich über die gesamte Gestaltung der Benutzeroberfläche (UI) und der Benutzererfahrung (UX). Angefangen bei der tiefgehenden Analyse der Nutzerbedürfnisse und -erwartungen bis hin zur Umsetzung eines intuitiven und ästhetisch ansprechenden Designs, trägt der UI/UX-Verantwortliche maßgeblich dazu bei, dass die Software nicht nur funktional, sondern auch benutzerfreundlich und effizient ist.

Benutzerzentrierter Entwurf: Der UI/UX-Verantwortliche sollte sich auf die Bedürfnisse und Erwartungen der Benutzer konzentrieren. Das bedeutet sicherzustellen, dass das Design den Anforderungen der Zielgruppe entspricht.

Informationsarchitektur: Verantwortlich für die Strukturierung und Organisation von Informationen innerhalb der Benutzeroberfläche. Dies umfasst die Gestaltung von Navigationsstrukturen, Menüs und Informationsflüssen, um eine intuitive Nutzung zu ermöglichen.

Prototyping und Wireframing: Erstellung von Prototypen und Wireframes, um das geplante Design zu visualisieren und frühzeitig Feedback zu sammeln. Dies erleichtert die Kommunikation mit dem Entwicklungsteam und anderen Stakeholdern.

Gestaltung von Benutzeroberflächen: Entwicklung von visuellen Designs für die Benutzeroberfläche, unter Berücksichtigung von Farben, Schriften, Bildern und anderen

grafischen Elementen. Dabei sollten Designprinzipien wie Konsistenz, Kontrast und Benutzerfreundlichkeit beachtet werden.

Usability-Tests: Durchführung von Tests, um die Benutzerfreundlichkeit und Effektivität der Benutzeroberfläche zu bewerten. Hierbei werden reale Benutzer in den Entwicklungsprozess einbezogen, um Feedback zu sammeln und Anpassungen vorzunehmen.

Zusammenarbeit mit Entwicklungsteam: Enge Zusammenarbeit mit den anderen Entwicklern, um sicherzustellen, dass das Design effizient und korrekt implementiert wird. Klärung von Fragen und Anpassung des Designs während des Entwicklungsprozesses.

Integration von Feedback: Aufnahme von Feedback aus verschiedenen Quellen, einschließlich Benutzertests, Stakeholder-Meinungen und Analysen, um kontinuierliche Verbesserungen an der Benutzeroberfläche vorzunehmen.

Technischer Architekt

Technische Architekten verwalten die Programm Infrastruktur des Projekts, z.B. Ordnerstruktur und Abhängige JAR Bibliotheken der Software (Dependencies). Zum Expertenwissen der Technischen Architekten zählen eindeutig sehr gute Kenntnisse der verwendeten Programmiersprachen, Toolkits und Frameworks, sowie Architekturelles Design und Software Patterns. Spezifische Horizontale Verantwortlichkeiten für dieses Projekt sind:

Projekt Objekt Model (Maven pom.xml):

- **Dependency Management:** Gewährleistung einer effizienten Verwaltung von Abhängigkeiten, insbesondere Versionen, im Maven-Projektobjektmodell (pom.xml).
- **Plugins:** Konfiguration und Integration von Plugins wie Checkstyle überprüfen, um die Qualität des Codes und der Ressourcen sicherzustellen.
- **Externe Maven Repositories:** Verwaltung der Verbindungen zu externen Maven-Repositories, um benötigte Bibliotheken und Ressourcen zu beziehen.

Expertenwissen zu allen verwendeten Technologien: Der technische Architekt muss über umfassendes Fachwissen zu allen eingesetzten Technologien verfügen, darunter Eclipse und Maven. Dies ist notwendig, um die Entwicklungsprozesse zu optimieren und sicherzustellen, dass die gewählten Werkzeuge effizient genutzt werden.

Erstellung von Checkstyle Guidelines: Der technische Architekt ist für die Definition von klaren Richtlinien für die Kodierung zuständig, um konsistenten Code sicherzustellen. Des Weiteren muss er die Integration von Checkstyle zur automatischen Überprüfung des Codes gemäß den definierten Richtlinien sicherstellen.

Erstellung von Richtlinien zur Sourcecode-Dokumentation (Javadoc): Festlegung von Standards für die Dokumentation des Sourcecodes durch Javadoc-Kommentare. Sicherstellung, dass der Code klar, verständlich und gut dokumentiert ist, um die Wartbarkeit zu verbessern.

Design der Programmarchitektur und -komponenten: Erstellung von UML-Komponentendiagrammen auf der Schnittstellenebene, um die Struktur und Interaktion der Softwarekomponenten zu visualisieren. Entwicklung von Verteilungsdiagrammen, um die Verteilung von Komponenten und deren Interaktionen in einer verteilten Umgebung zu modellieren.

Projektplan

Work Breakdown Structure

Die WBS wurde auf Meilenstein-Ebene entworfen und es wurden die ersten Arbeitspakete aufgrund der Horizontalen Verantwortungen berücksichtigt. Die technischen Arbeitspakete der Iceberglist werden in der WBS in diesem Projekt nur auf Meilensteinebene berücksichtigt - da wir technisch nach SCRUM vorgehen. Das heisst es werden hier nur "nicht-technische" Arbeitspakete (Dokumentation, Konfiguration, etc.) eingetragen.

Nr	Arbeitspaket	Anfang	Ende	Verantwortlich
MS.0	Kick-Off	07.11.2023	08.11.2023	All
1.1	Anforderungsanalyse	10.11.2023	14.11.2023	Pjanic
1.1.1	Feature-Liste verfeinern: Iceberglist	10.11.2023	14.11.2023	Pjanic
1.1.2	Projektplan fertigstellen	10.11.2023	14.11.2023	All
1.1.3	Rollenverteilung ausarbeiten	11.11.2023	14.11.2023	All
1.1.4	Template und Stundenliste in Git hinzufügen	11.11.2023	14.11.2023	Epner
MS.1	Projektdefinition, Projektauftrag	13.11.2023	14.11.2023	Kleinlehner
MR-1	Management Review 1	15.11.2023	15.11.2023	Kleinlehner
2.1	UI Sketch anfertigen	15.11.2023	23.11.2023	Epner
2.2	Komponentendiagramm anfertigen	15.11.2023	21.11.2023	Gobl
3.1	Implementierung Sprint 1 (MS.2)	15.11.2023	23.11.2023	All
3.2	ER Diagram erstellen	15.11.2023	23.11.2023	Gobl
3.3	Testdaten verfügbar machen	15.11.2023	23.11.2023	Eckermann
3.4	Persistenzschicht testen	15.11.2023	23.11.2023	Eckermann
3.5	Erste GUI Panels	15.11.2023	23.11.2023	Epner
MS.2	Erster Systemtest	23.11.2023	23.11.2023	Eckermann
IR-1	Internes Review	27.11.2023	01.12.2023	Kleinlehner
3.6	Implementierung Sprint 2 (MS.3)	30.11.2023	12.12.2023	Kleinlehner
MS.3	GUI Prototyp mit DAO Tests, 60% der User Stories implementiert	12.12.2023	12.12.2023	All

MR-2	Management Review 2 (Design Review), Alpha Release Demo	12.12.2023	12.12.2023	Kleinlehner
3.7	Implementierung Sprint 3 (MS.4)	19.12.2023	09.01.2023	Kleinlehner
MS.4	80% der User Stories implementiert, Service-Schicht Tests	09.01.2023	09.01.2023	All
IR-2	Internes Review (Code Inspektion)	09.01.2023	09.01.2023	Kleinlehner
3.8	Implementierung Sprint 4 (MS.5)	09.01.2023	22.01.2023	Kleinlehner
MS.5	100% der User Stories implementiert, Akzeptanztests	22.01.2023	22.01.2023	All
MR-3	Managment Review3: Projektabnahme, General Availability Demo	22.01.2023	25.01.2023	Kleinlehner

Informationswesen

Die Informationsstruktur für das Projekt wird folgendermaßen aussehen:

- Wöchentliche Treffen mit dem Tutor (Jour-Fixe)
- Insgesamt 5 Reviews (2 IRs, 3 MRs) zu den 5 Meilensteinen mit 5 Statusberichten
- Elektronische Kommunikation synchron mittels Discord und Google Docs, asynchron mittels Gitlab. Die Mailingliste ist mit niedrigerer Priorität zu verwenden.
- Kommunikation mit den Auftraggebern und Tutor per Mail

Die zur Verfügung gestellte Infrastruktur umfasst:

- Gitlab Repository (SVN + CI/CD Pipeline)

USP

Besonderheiten bei diesem Projekt sind vor allem bei den Cocktail Vorschlägen und den Vorschlägen welche Zutaten gekauft werden sollen. Bei beiden Problemen handelt es sich um Optimierungsprobleme. Um den Ablauf klar zu definieren, wird ein Beispiel angeführt, welches das Problem darstellt und etwaige Unklarheiten beseitigt.

Der User A ist derzeit Host und möchte eine Cocktailkarte erstellen. Die Gruppe hat ihre Präferenzen und Zutaten bereits bestimmt. Somit kann A mit dem Erstellprozess beginnen. Im ersten Schritt wird A vorgeschlagen, welche Zutaten er extra kaufen kann, um weitere Cocktails zu machen. Die Zutaten Vorschläge sind dabei so gewählt, dass die Auswahl an neuen Cocktails maximal ist -> Maximierung. Im zweiten Schritt kann A manuell Cocktails aus der Liste der möglichen Cocktails zur Karte hinzufügen, oder er nutzt das Automatische erstellen feature. Dieses probiert anhand der Geschmacks Prioritäten und der Einstellung anderer Parameter eine optimale Karte zu erstellen -> Optimierung. Diese Karte kann anschließend noch manuell verändert werden. Das Ergebnis ist am Ende eine Cocktailkarte und eine Übersicht welche Zutaten gekauft werden müssen, um diese anbieten zu können.

Risikoanalyse

Nr	Typ	Auswirkungen	Eintrittswahrscheinlichkeit	Folgeriskiken	Verantwortliche	Name & Beschreibung	Gegenmaßnahme
1	immer	hoch	mittel	2	kleinlehner	**Projektmiglied fällt aus:** Ein Projektmitglied kann, zB bedingt durch eine Krankheit, längere Zeit nicht am Projekt mitarbeiten bzw scheidet komplett aus dem Projektteam aus.	Sammeln der bereits geleisteten und noch ausständigen Arbeit; Besprechung und Aufteilung der Aufgaben innerhalb der Gruppe (zusätzlich zum Jour-Fixe, kann auch elektronisch stattfinden). Zuteilung der Rolle zu einem anderen Gruppenmitglied.
2	entwicklung	hoch	niedrig			**Rechtzeitige Fertigstellung gefährdet:** Die rechtzeitige Fertigstellung des Projekts ist gefährdet. Mögliche Ursachen könnten zu hohe Anforderungen, zu geringe zeitliche Ressourcen, Ausfall von Team-Mitgliedern, etc sein.	Reduktion von Features anhand ihrer Priorisierung (abhängig von aktuellem Fortschritt)
3	entwicklung	hoch	niedrig			**Maximierungsproblem nicht schnell genug lösbar**: Es stellt sich durch Tests heraus, dass das komplexe Problem nicht mit der gewünschten Genauigkeit in wenigen Sekunden gelöst werden kann.	Höhere Wartezeit in Kauf nehmen und den User darüber informieren.
4	planung	niedrig	mittel	2	epner	**Fehler im Front-End Design:** Durch einen Fehler im Front-End Design wird die Benutzeroberfläche unverständlich und erschwert dem Benutzer die Bedienung.	Design Entwürfe im Team besprechen und wenn nötig neu ausarbeiten.