M223: Multiuser Applikation

Probe IPA

Erstellt durch: Lara Felix

1 Inhaltsverzeichnis

2 Vorwort

Dieses Projekt wird im Rahmen des ÜK's M223 durchgeführt. Es wird als Probe IPA strukturiert.

Die Aufgabe dieses Projektes ist es, ein objektorientierter Multi – User – Applikation zu erstellen. Das bedeutet, es sollen mehrere User gleichzeitig auf den gleichen Datenbestand zugreifen können. Zudem soll es verschieden Benutzer und Rechte geben.

3 Organisation der Arbeitsergebnisse

Der Stand des Projekts wird mindestens nach jedem Tag auf Git abgelegt. Vorzugsweise wird die Arbeit auch nach jeder grösseren Änderung, zum Beispiel beim Erreichen eines Meilensteins, abgelegt. In der Versionentabelle unten wird nur die abgelegte Version eines Tages (Ende des Tages) festgehalten. Die anderen Versionen, können auf Git eingesehen werden.

Tabelle 1 Versionierung

Version	Datum	Änderung	Autor
1.0	08.12.2021	• Arbeitsjournal Tag 6 • Überarbeiten ganze Dokumentation • Reflexion • Anhang • Quellen- /Abbildungs- /Tabellen – Verzeichnis • Auswertung erstellt	LF
0.8	07.12.2021	• Realisieren erstellt • Testing durchgeführt • Arbeitsjournal Tag 5	LF
0.6	06.12.2021	• Arbeitsjournal Tag 4	LF
0.4	01.12.2021	• Planen erweitern • Use Case in Textform • Testing erstellt • Arbeitsjournal Tag 3	LF
0.2	30.11.2021	• UML's (Use Case, Wireframe) • Ddl erstellt • Informieren, Planen und Entscheiden erstellt • Arbeitsjournal Tag 2	LF
0.0	29.11.2021	• Dokumentation inkl. Struktur erstellt • Gantt Planung • Arbeitsjournal Tag 1	LF

4 Teil 1 – Umfeld und Ablauf

4.1 Aufgabenstellung

Im Rahmen des ÜK Moduls 223 soll eine Multiuser Applikation erstellt werden.

Folgende Rahmenbedingungen wurde durch den Modulleiter Remo Steinmann vorgegeben.

- Objektorientierte Multi-User-Applikation
- Umfang gemäss Planung (ca. 3 Tage Entwicklung, 2.5 Tage Dokumentation; 6h pro Tag)
- Zentrale Datenbank
- Mehrere Clients müssen gleichzeitig auf den gleichen Datenbestand zugreifen
- Zentrale Benutzer- und Rechte-Verwaltung

Für dies werde ich das bekannte Spiel «Faltpapier-Geschichte» digitalisieren. Dieses Spiel funktioniert wie folgt:

1. Spieler 1 schreibt zwei Sätze
2. Spieler 2 sieht den zweiten Satz des Spieler 1
3. Aufgrund dessen schreibt er ebenfalls zwei Sätze
4. Spieler 3 sieht nur den zweiten Satz von Spieler 2
5. Aufgrund dessen schreibt er ebenfalls zwei Sätze

Und so weiter.

Das Erstellen der zwei Sätze wird Zeitbegrenzt sein, sodass man nicht zu lange studiert und die Geschichte spontaner wird. Zudem kann jeder User nur 1-mal alle 2h die Geschichte um zwei Sätze erweitern. So schreibt der User nicht eine Geschichte allein und es schützt vor Spam-Angriffen. Nach dem Absenden der erstellten Sätze, gibt es die Möglichkeit den aktuellen Stand Geschichte herunterzuladen.

Es gibt folgende Rollen und ihre Berechtigungen.

- **Writer**
  - Standardrolle, wenn man sich neu registriert.
  - Einloggen und ausloggen
  - Schrieben von 2 Sätzen
  - Herunterladen der Geschichte
- **Game Master**
  - Wird in der DB manuell von einem Writer zu einem Game Master migriert
  - Einloggen und ausloggen
  - Sieht aktueller Stand der Geschichte
  - Sieht welcher User was Geschrieben hat

Alle User müssen sich beim Start der Applikation über ein Login Fenster authentifizieren. Die Authentifizierung wird mittels Mailadresse und Passwort durchgeführt. Es gibt keine weiteren Möglichkeiten der Authentifizierung. Falls der User noch kein Login besitz, kann er sich über den Registrierungs-Button als neuen User anmelden. Es ist nicht möglich, als einen Gast mitzuschreiben.

Bei den Userdaten werden nur die Mailadresse sowie das Passwort in der Datenbank gespeichert.

Es werden für die gesamte Projektdokumentation die Standards des Betriebs bzw. der Abteilung verwendet. So wie die Standards Programmiersprachen der Abteilung (Perl und Java). Alle Daten werden in einer zentralen relationalen MySQL-Datenbank abgespeichert.

Folgendes wird in diesem Projekt nicht berücksichtigt.

- Keine Profiländerungen des Benutzers
- Änderungen am Datenbestand durch einen anderen Client wird nicht automatisiert im eigenen Programm aktualisiert
- Die Applikation muss nur auf dem Gerät, welches während der Arbeit benutzt wird, laufen.
- Andere Betriebssysteme oder Geräte werden nicht berücksichtigt.
- Die Versionierung wird nur mittels Gitlab gestattet.

4.2 Projektaufbauorganisation

Auftraggeber

Remo Steinmann

remo.steinmann@siemens.com

Auftragnehmer

Lara Felix

lara.felix@siemens.com

078 633 50 99

Experte

Remo Steinmann

remo.steinmann@siemens.com

Nebenexperte

Ruwen Wiederkehr

ruwen.wiederkehr@siemens.com

Nebenexperte

Merjem Hamza

merjem.hamza@siemens.com

4.3 Mittel und Methoden

Folgende Mittel werden in diesem Projekt gebraucht:

- Programmiersprache: Java, Perl, SQL
- Datenbank: MySQL
- Betriebssystem: Windows 10
- Versionierungssystem: GitHub Repository und Git
- Diagramme (UML's): diagrams.net
- DIE: Visual Studio Code

Als Projektmanagementmethode wird IPERKA benutzt.

4.4 Vorkenntnisse

Ich arbeite in meiner Abteilung hauptsächlich mit den Programmiersprachen Java und Perl. Das verbinden von Frontend (Java) und Backend (Perl) ist mir ebenfalls durch die Arbeit in meiner Abteilung bekannt. Jedoch stehen dort diverse Template für das Frontend mittels Java zur Verfügung. Alles vom Grunde auf neu aufzubauen wird für mich Neuland sein. Das Einbinden einer Datenbank ist mir ebenfalls schon bekannt, dies jedoch auch wieder nur mittels Librarys.

Eine Multiuserapplikation an sich habe ich noch nie erstellt.

Die oben aufgeführten Mittel habe ich alle schon mehrmals eingesetzt.

4.5 Vorarbeit

Für das Erstellen des Projektantrags, machte ich mir schon ein paar Gedanken bezüglich des groben Aufbaus. Dies wird im Schritt Planen erwähnt.

4.6 Deklaration der benützten Firmenstandards

Es wird die aktuelle Version der Firmenstandards Siemens Mobility benutzt

4.7 Zeitplan

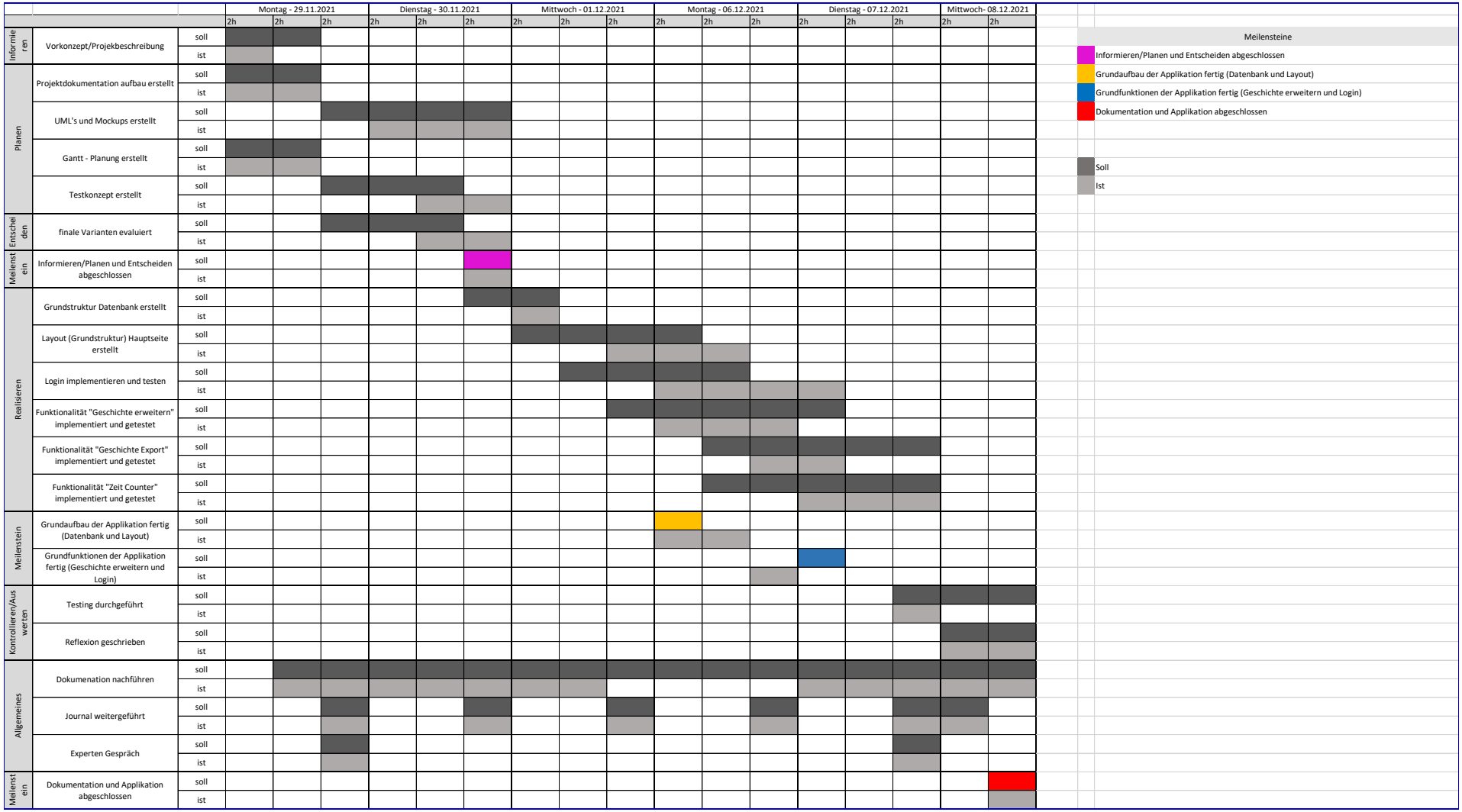


Abbildung 1: Gantt-Planung

4.7.1 Meilensteine

Tabelle 2 Meilensteine

Meilenstein	Zeitpunkt - Soll	Status / Ist
Informieren, Planen und Entscheiden abgeschlossen	30.11.2021 - 17:00 Uhr	OK 01.12.2021 – 11.00 Uhr
Grundaufbau der Applikation fertig (Datenbank und Layout)	06.12.2021 - 10:00 Uhr	OK 06.12.2021 – 11.00 Uhr
Grundfunktionen der Applikation fertig (Geschichte erweitern und Login)	07.12.2021 - 10:00 Uhr	OK 06.12.2021 – 16.00 Uhr
Dokumentation und Applikation abgeschlossen	08.12.2021 – 12.00 Uhr	OK 07.12.2021 – 12-00 Uhr

4.8 Arbeitsprotokoll

4.8.1 29.11.2021 – Tag 1

Geplant

- Projektdokumentation Aufbau erstellen
- Gantt - Planung erstellen
- Projektjournal Tag 1
- Vorkonzept / Projektbeschreibung
- Gespräch mit HE (Remo Steinmann)

Gemacht

- Gantt – Planung
- Projektdokumentation Aufbau
- Projektjournal Tag 1
- Einlesen in FArbeit_2021 und Kriterienkatalog
- Aufgabenstellung
- Projektaufbauorganisation
- Erstes Gespräch mit Hauptexperten (Remo Steinmann)

Probleme

Bei der eigentlichen Umsetzung von den oben genannten Punkten traten keine Probleme auf. Jedoch kam ich nicht so gut voran wie gedacht.

Fazit

Folgendes wurde im Gespräch besprochen

- Gantt Planung
  - Schritte mehr aufteilen (z.B. Login planen, Login erstellen, Login testen)
  - Legende im Excel – File und nicht nur in Textform in der Dokumentation
  - Meilensteine Soll/Ist Vergleich zum Beispiel Meilenstein als eigenen Task

    - Jeden Iperka – Schritt aufteilen und nicht zusammenfassen
    - Schriftgrösse von Tasks grösser → Soll/Ist – Boxen im Vergleich zu gross
- Aufgabenstellung
    - Gut, kommt von abgegebener Aufgabenstellung umgeändert in die Dokumentation
- Projektdokumentation Aufbau
    - Glossar, Quellenverzeichnis, Abbildungsverzeichnis und Tabellenverzeichnis kommen in den Punkt Anhang und wird als Punkt 6 gekennzeichnet

Die oben genannten Verbesserungspunkte werden bis morgen Abend 30.11.2021 17.00 Uhr angepasst.

4.8.2 30.11.2021 – Tag 2

Geplant

- UML's Mockup
- Testkonzept
- Projektjournal Tag 2
- Varianten evaluieren
- Dokumentation nachführen
    - Teil 1
    - Informieren und Planen Teil 2

Gemacht

- Teil 1 nachgeführt
    - Mittel und Methoden
    - Vorkenntnisse
    - Vorarbeit
- Wireframes
- Ddl
- Use Case (muss noch überarbeitet werden)
- Projektjournal Tag 2

Probleme

Ich konnte mich, wie gestern, nicht so gut konzentrieren. Da wir uns in dieser Konstellation schon lange nicht mehr gesehen haben. Deshalb wurde viel geredet anstatt gearbeitet. Ich hoffe, dass ich dies morgen im Home Office aufholen kann. Falls dies nächste Woche immer noch der Fall sein sollte, werde ich mehr Musik hören, um konzentrierter zu arbeiten und nicht abgelenkt zu werden. Bei dem Use Case war ich mir wegen den Notationen nicht mehr sicher. Dies klärte ich jedoch mit Remo.

Gelerntes

Input Remo

- Check-Liste für Gespräch mit HEX
    - Aufgeräumt
    - Infrastruktur geprüft (vor ort bzw. zu Hause)
    - HEX als Besucher anmelden
    - Gantt - Planung parat
    - Geht ca. 20 min.
    - Kriterienkatalog ausdrucken
    - Tipps umsetzen
- Use Case
    - Überblick über das Programm geben
    - Muss noch in Text erläutert werden
    - Akteur, Voraussetzung, Kurz Beschreib

Es ist wichtig, dass ich fokussierter arbeite. Dies ist mein Hauptproblem. Ich konnte viel aus diesem Tag mitnehmen insbesondere die Inputs von Remo haben mir auf Hinsicht dieses Projektes sowie auf die IPA sehr geholfen.

Konnte nicht umgesetzt werden und muss morgen nachgeholt werden

- Use Case überarbeiten mit Inputs von Remo
- Testkonzept (Können von Use Case abgeleitet werden)

- Evtl. weitere UML's einfügen bzw. bestehende überarbeiten.

4.8.3 01.12.2021 – Tag 3

Geplant

- Nachholen von Gestern
  - Use Case überarbeiten mit Inputs von Remo
  - Testkonzept (Können von Use Case abgeleitet werden)
  - Evtl. weitere UML's einfügen bzw. bestehende überarbeiten.
- Grundstruktur Datenbank erstellt (muss heute fertig sein)
- Layout (Grundstruktur) Hauptseite erstellen
- Login implementieren anfangen
- Funktionalität "Geschichte erweitern" anfangen
- Doku nachführen
- Projektjournal Tag 3
- Zwischengespräche mit Remo

Gemacht

- Projektjournal Tag 3
- Use Case in Textform
- Testkonzept (Können von Use Case abgeleitet werden)

Probleme

Ich konnte mich, wie gestern, nicht so gut konzentrieren. Da wir uns in dieser Konstellation schon lange nicht mehr gesehen haben. Deshalb wurde viel geredet anstatt gearbeitet. Ich hoffe, dass ich dies morgen im Home Office aufholen kann. Falls dies nächste Woche immer noch der Fall sein sollte, werde ich mehr Musik hören, um konzentrierter zu arbeiten und nicht abgelenkt zu werden. Bei dem Use Case war ich mir wegen den Notationen nicht mehr sicher. Dies klärte ich jedoch mit Remo.

Gelerntes

Input Remo

- Arbeitsjournal
  - Planung innerhalb des Tages mit Zeit angaben

Es ist wichtig, dass ich fokussierter arbeite. Dies ist mein Hauptproblem. Ich konnte viel aus diesem Tag mitnehmen insbesondere die Inputs von Remo haben mir auf Hinsicht dieses Projektes sowie auf die IPA sehr geholfen.

Konnte nicht umgesetzt werden und muss morgen nachgeholt werden

- Use Case überarbeiten mit Inputs von Remo
- Testkonzept (Können von Use Case abgeleitet werden)
- Evtl. weitere UML's einfügen bzw. bestehende überarbeiten.

4.8.4 06.12.2021 – Tag 4

Geplant

- Grundaufbau der Applikation fertig (Datenbank und Layout)
- Projektjournal Tag 4
- Verbindung mit DB fixen
- Dokumentation erweitern

Tabelle 3 Planung Tag 4

Zeit	Was	Status
8.30 – 9.00	Projektjournal Tag 4 erstellen (Geplant)	OK
9.00 – 11.00	Grundaufbau der Applikation fertig (Datenbank und Layout)	OK
11.00 – 12.00	Dokumentation erweitern	OK
13.00 – 14.00	Verbindung mit DB fixen	OK
14.00 – 16.00	Funktionalität "Geschichte erweitern" implementieren	OK
16.00 – 16.30	Dokumentation nachführen und Projektjournal Tag 4 abschliessen	OK

Gemacht

- Projektjournal Tag 4
- Grundaufbau der Applikation fertig (Login/registration, expand und download)
- Verbindung mit DB gefixt

Probleme

Beim Programmieren traten keine grosse Probleme auf. Wie ich die Zwischentabelle tab_userstory richtig fülle, weiss ich noch nicht genau bzw. ist noch nicht fertig implementiert. Als erstes dachte ich, es wäre eine gute Idee, sie via einem Trigger zu füllen, welcher bei einem neuen Eintrag in die Story-Tabelle ausgelöst wird. Dort habe ich jedoch das Problem, dass ich nicht weiss, welcher User diesen neuen Eintrag erstellt hat. Deshalb wird der Eintrag in die tab_userstory gleichzeit mit dem Eintrag in die tab_story erstellt. Ich habe heute eigentlich keine Zeit für die Dokumentation verwendet, jedoch bin ich gut bei der Applikation vorwärts gekommen, deshalb ist dies oke. Ich werde morgen mehr Zeit für die Dokumentation aufwenden um vor allem auch den Realisieren – Teil anzufangen, denn dieser ist momentan noch ziemlich mager.

Gelerntes

Ich konnte heute meine Java-Kenntnisse auffrischen. Ich arbeite zwar in meiner Abteilung mit Java, jedoch sind dort schon viele Templates und Librarys zur Verfügung. Das ganze also von Grunde auf neu aufzubauen, habe ich schon seit längerer Zeit nicht mehr gemacht.

Ich konnte heute viel fokussierter Arbeiten als die letzten Tage. Ich denke, dass liegt zum einen daran, dass die Anderen auch nicht so viel geredet haben und zum anderen habe ich die meiste Zeit Musik gehört. Die Kopfhörer habe ich auf «Geräuschunterdrückung» eingestellt. So konnte ich mich noch mehr fokussieren. Ich werde dies für den morgigen Tag auf jeden Fall übernehmen, sodass ich fokussierter Arbeiten kann.

Konnte nicht umgesetzt werden und muss morgen nachgeholt werden

- Arbeitsjournal Tag 4 nochmals überarbeiten
- Dokumentation erweitern

4.8.5 07.12.2021 – Tag 5

Geplant

- Dokumentation Realisieren-Teil
- Projektjournal Tag 5
- Projektjournal Tag 4 überarbeiten
- Gantt Planung nachführen
- Funktion Zeit-Counter
- Tab_userstory abfüllen
- Inputs validieren
- Dokumentation überarbeiten
- Testing durchführen
- Gespräch mit HEX (Remo Steinmann)

Tabelle 4 Planung Tag 5

Zeit	Was	Status
7.30 – 8.00	Projektjournal Tag 5 erstellen (Geplant), Projektjournal 4 überarbeiten und Gantt-Planung nachführen	OK
8.30 – 11.00	Tab_userstory abfüllen, Funktion Zeit-Counter	OK
11.00 – 12.00	Inputs validieren	OK
13.00 – 14.00	Dokumentation Realisieren - Teil	OK
14.00 – 16.00	Gespräch mit HEX, Dokumentation Realisieren – Teil, Testing durchführen	OK
16.00 – 16.30	Dokumentation überarbeiten und Projektjournal Tag 5 abschliessen	OK

Gemacht

- Projektjournal Tag 5
- Projektjournal Tag 4 überarbeiten
- Gantt Planung nachführen
- Tab_userstory abfüllen
- Gespräch HEX (Remo Steinmann)
    - Versionierung – Änderungen nur Änderung von Dokumentation protokollieren
    - Planung nur einmal drin
    - Spezifische Tagesplanung bei Arbeitsjournal bei den vorherigen Tagen nicht im Nachhinein einfügen.
    - Für Morgen Adapter mitnehmen. Testen ob es mit Biemer funktioniert für Präsentation
- Inputs validieren
- Dokumentation Realisieren Teil
- Testing durchgeführt

Probleme

Ich konnte heute fast alles erledigen, was ich geplant habe. Leider nahm das abfüllen der Tab_UserStory sowie die Funktionen des Zeit Counters mehr Zeit ein als geplant. Deshalb musste ich am Abend zu Hause noch die Dokumentation ein wenig überarbeiten, sodass ich am letzten Tag, also morgen, keinen Stress habe und genügend Zeit habe, alles nochmals anzuschauen.

Die Änderungstipps, welche ich von Remo im Gespräch erhalten habe, setzte ich ebenfalls noch nach dem Gespräch um. So brauchte ich noch mehr ungeplante Zeit. Beim Gespräch ging der Anschluss an

den Bildschirm nicht. Für das muss ich morgen noch meine Adapter von zu Hause aus mit nehmen, in der Hoffnung das der Funktioniert.

Gelerntes

Ich konnte heute erneut meine Java-Kenntnisse auffrischen. Zudem brauchte ich heute seit langem wieder mal Regex in Java. Dies bringt mir im Hinblick auf die IPA viel, da ich dort sehr wahrscheinlich auch den Input in den Feldern überprüfen muss und dies ebenfalls mit Regex machen werden.

4.8.6 07.12.2021 – Tag 6

Geplant

- Dokumentation abschließen
  - Auswerten
  - Reflexion
  - Glossar
  - Anhang
  - Quellen- /Abbildungs- /Tabellen – Verzeichnis
  - Gantt- Planung nachführen
- Projektjournal Tag 6
- Alles überarbeiten (mit Kriterienkatalog vergleichen/überprüfen)

Tabelle 5 Planung Tag 6

Zeit	Was	Status
7.00 – 8.00	Projektjournal Tag 6 erstellen (Geplant), Abbildungsverzeichnis	OK
8.00 –10.00	Auswerten inkl. Reflexion, Gantt – Planung nachführen	OK
10.00 -11.00	Glossar, Anhang, Quellen- /Abbildungs- /Tabellen – Verzeichnis	OK
11.00 – 12.00	Alles überarbeiten (mit Kriterienkatalog vergleichen/überprüfen)	OK
12.00	Abgabe	OK

Gemacht

- Dokumentation abschließen
  - Auswerten
  - Reflexion
  - Glossar
  - Anhang
  - Quellen- /Abbildungs- /Tabellen – Verzeichnis
  - Gantt- Planung nachführen
- Projektjournal Tag 6
- Alles überarbeiten (mit Kriterienkatalog vergleichen/überprüfen)

Probleme

Es gab heute keine grossen Probleme. Ich hätte jedoch mit etwas mehr Zeit noch einiges schöner machen können. Jedoch fehlte am Schluss für das Zeit und Motivation. Ich überarbeitete jedoch trotzdem noch einige Teile und kontrollierte meine Arbeit mit dem Kriterienkatalog. Dort ist jedoch alles ziemlich offen formuliert und so Erfüllte meine Arbeit alle Kriterien (meiner Meinung nach). Es gibt immer noch überall kleine Baustellen, welche ich gerne erledigt hätte, da ich doch gewisse Ansprüche an meine Leistung habe. Wenn ich am Anfang des Projektes besser und vor allem genauer gearbeitet hätte, gäbe es schon einige Baustellen vor allem in der Dokumentation weniger. Dies muss ich auf jeden Fall bei einem nächsten Projekt besser machen. Ansonsten ist mein Ergebnis gut.

Gelerntes

Ich konnte in diesem Projekt vieles Lernen. Ich konnte mein veraltetes Wissen bezüglich Java und SQL-Abfragen auffrischen. Zudem konnte ich vor allem auch viel Wissen sammeln, welches mir bei der IPA helfen wird. So zum Beispiel die Tipps von Remo oder der Aufbau der Dokumentation.

5 Teil 2

5.1 Kurzzusammenfassung

Dieses Projekt wird im Rahmen des ÜK's M223 durchgeführt. Es wird als Probe IPA strukturiert.

Die Aufgabe dieses Projektes ist es, ein objektorientierter Multi – User – Applikation zu erstellen. Das bedeutet, es sollen mehrere User gleichzeitig auf den gleichen Datenbestand zugreifen können. Zudem soll es verschieden Benutzer und Rechte geben.

Um diese Kriterien zu erfüllen, digitalisiere ich das bekannte Spiel Faltpapiergeschichte. Die Applikation wird mit Java und MySQL umgesetzt. Es gibt zwei Rollen. Ein Writer, welcher die Geschichte erweitern kann und sie auch herunterladen kann. Zudem gibt es noch einen Game Master, welcher alles kann, was ein normaler Writer kann, jedoch sieht er, welcher User, welche Storyline erfasst hat.

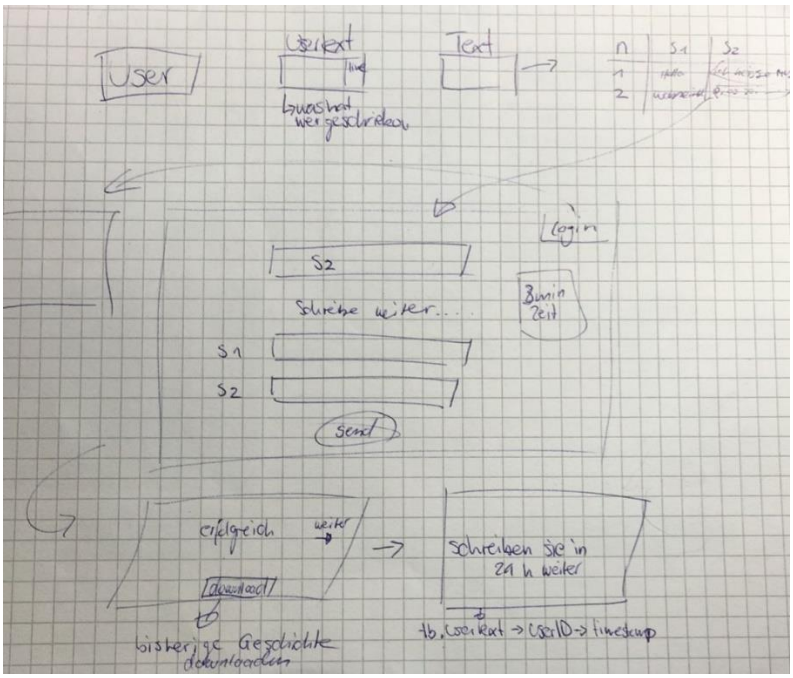
Es konnte alles wie geplant umgesetzt werden und alle Kriterien werden erfüllt. Es könnte jedoch einiges besser gemacht werden. So hätte man das Layout der Applikation schöner erstellen können, einiges im Code auslagern, sodass es keinen redundanten Code gibt und alles hätte ein wenig detaillierter Dokumentiert werden können.

5.2 Informieren

Folgende Rahmenbedingungen wurde durch den Modulleiter Remo Steinmann vorgegeben

- Objektorientierte Multi-User-Applikation.
- Umfang gemäss Planung (ca. 3 Tage Entwicklung, 2.5 Tage Dokumentation; 6h pro Tag)
- Zentrale Datenbank
- Mehrere Clients müssen gleichzeitig auf den gleichen Datenbestand zugreifen
- Zentrale Benutzer- und Rechte-Verwaltung

Um dies zu erreichen digitalisiere ich das bekannte Spiel Papiergeschichte schreiben. Ich machte mir einige Gedanken bezüglich des Aufbaus. Dies hielt ich auf einer groben Skizze fest.



5.2.1 Rollen

Es gibt folgende Rollen mit ihren Autorisierungen.

- **Writer**
  - Standardrolle, wenn man sich neu registriert.
  - Einloggen und ausloggen
  - Schreiben von 2 Sätzen
  - Herunterladen der Geschichte
- **Game Master**
  - Wird in der DB manuell von einem Writer zu einem Game Master migriert
  - Einloggen und ausloggen
  - Sieht aktueller Stand der Geschichte
  - Sieht welcher User was Geschrieben hat

Alle User müssen sich beim Start der Applikation über ein Login Fenster authentifizieren. Die Authentifizierung wird mittels Mailadresse und Passwort durchgeführt. Es gibt keine weiteren Möglichkeiten der Authentifizierung. Falls der User noch kein Login besitz, kann er sich über den Registrierungs-Button als neuen User anmelden. Es ist nicht möglich, als einen Gast mitzuschreiben.

5.2.2 Wertebereich

Es werden folgende Wertebereiche festgelegt. Zudem soll die Eingabe bzw. das Absenden von keiner Eingabe nicht möglich sein. Falls die E- Mail Adresse bereits gebraucht wurde, soll dies ebenfalls mit einem Fehlermeldung an den User mitgeteilt werden.

Tabelle 6 Wertebereich

Was	Wertebereich
E-Mail Adresse	muss @ und . beinhalten
Passwort	min. 8 Zeichen
Satz 1	0-25 Zeichen, Buchstabe, Zahlen, Sonderzeichen (. , ! ? -)
Satz 2	0-25 Zeichen, Buchstabe, Zahlen, Sonderzeichen (. , ! ? -)

5.3 Planen

Im nächsten Schritt werden die Funktionen sowie der Aufbau der Applikation definiert.

5.3.1 Use Case

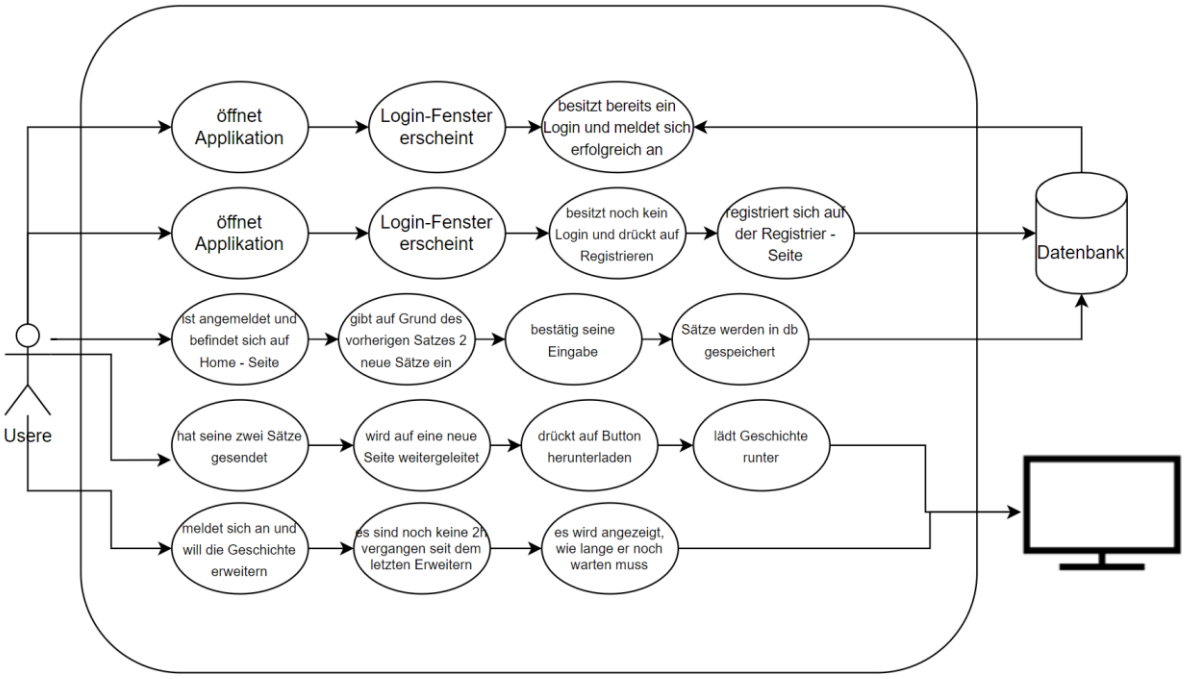


Abbildung 2 Use Case

5.3.2 Use Case Textform

Tabelle 7 Use Case Textform

UC1 --> Login	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User gibt korrektes Passwort und E- Mail – Adresse ein. Login – Informationen werden überprüft. Login – Daten sind korrekt. User wird zur Hauptseite weitergeleitet.
Nachbedingung	keine
UC2 --> Registrieren	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist noch nicht als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User drückt auf Registrieren – Button oben rechts. Es erscheint das Registrierung – Fenster. User gibt seine E-Mail Adresse und Passwort ein und bestätigt seine Eingabe via. «registrieren»-Button. User wird in der DB gespeichert. User wird zur Hauptseite weitergeleitet.
Nachbedingung	keine
UC3 --> -Geschichte erweitern	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und befindet sich auf der Hauptseite
Beschreibung	User sieht den letzten Satz der Geschichte. Schreibt in die vorgesehenen Inputfelder zwei neue Sätze aufgrund des letzten Satzes der Geschichte. Die Eingabe der neuen Sätze muss innerhalb 2h eingegeben werden. Die Eingabe wird mittels «senden»-Button bestätigt und in der Datenbank abgespeichert.
Nachbedingung	keine
UC4 --> Geschichte herunterladen	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und hat seine zwei Sätze abgesendet.

Beschreibung	User wird nach dem Absenden der zwei Sätze auf die Herunterladen – Seite weitergeleitet. Durch den Download - Button hat der User die Möglichkeit die Geschichte herunterzuladen.
Nachbedingung	keine
UC5 --> Geschichte erweitern bevor 2h vorbei sind	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert und hat innerhalb der letzten zwei Stunden die Geschichte bereits erweitert.
Beschreibung	Bevor der User nach dem Anmelden auf die Hauptseite weitergeleitet wird, wird überprüft, ob er innerhalb der letzten zwei Stunden die Geschichte bereits erweitert hat. Falls dies der Fall ist, wird er nicht auf die Hauptseite weitergeleitet. Es erscheint eine Seite mit der Zeit Angabe, wie lange der User warten muss, bis er weiterschreiben darf.
Nachbedingung	keine

5.3.3 Backend

Ursprünglich wäre das Backend mit Perl umgesetzt worden. Jedoch werde ich dies nicht machen, da es keine grossen Verarbeitungen der Daten gibt. Die Datenbank abfragen werden direkt aus dem Frontend mit Java gemacht.

5.3.3.1 Datenbank

Die Datenbank ist eine mySQL Datenbank und wird mittels xampp lokal gehostet.

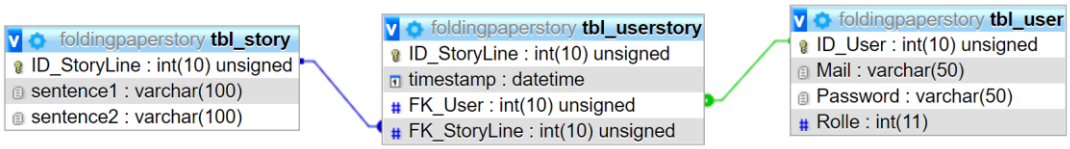


Abbildung 3 ERM

5.3.3.1.1 tbl_story

in der Tabelle tbl_story werden die Sätze der Geschichte abgespeichert. Der Datentyp beider Sätze ist Varchar(100). Zudem hat jede Storyline also zwei Sätze zusammen, erstellt von einem User, eine ID.

5.3.3.1.2 tbl_userstory

Die Tabelle Userstory wird gebrauch, um den User mit den Storylines zu verknüpfen. Ein User kann mehrere Storylines haben, eine Storyline aber nur einen User. Zudem wird ein timestamp erstellt, sobald ein User seine Sätze abgeschickt hat. Dieser wird gebraucht, um sicherzustellen, dass ein User nur alle zwei Stunden die Geschichte erweitern kann.

Dieser Aufbau der Datenbank wurde so gewählt, um Erweiterungen der Applikation zu ermöglichen.

5.3.3.1.3 tbl_user

In dieser Tabelle werden die Informationen eines Users gespeichert. Ein User besitzt eine eindeutige Userid, eine Mail Adresse, ein gehashtes Passwort und seine Rolle. Es gibt momentan die Rolle 0 für den Writer und eine Rolle 1 für den Gamemaster. Falls die Applikation erweitert wird, können weitere Rollen hinzugefügt werden.

5.3.4 Frontend

Das Frontend wird mittels Java erstellt und verwaltet alle Funktionalitäten der Applikation.

5.3.4.1 Wireframe

Die unten dargestellten Wireframes sind nach ihrer chronologischen Reihenfolge angeordnet und nummeriert. Das Design wird schlicht gehalten.

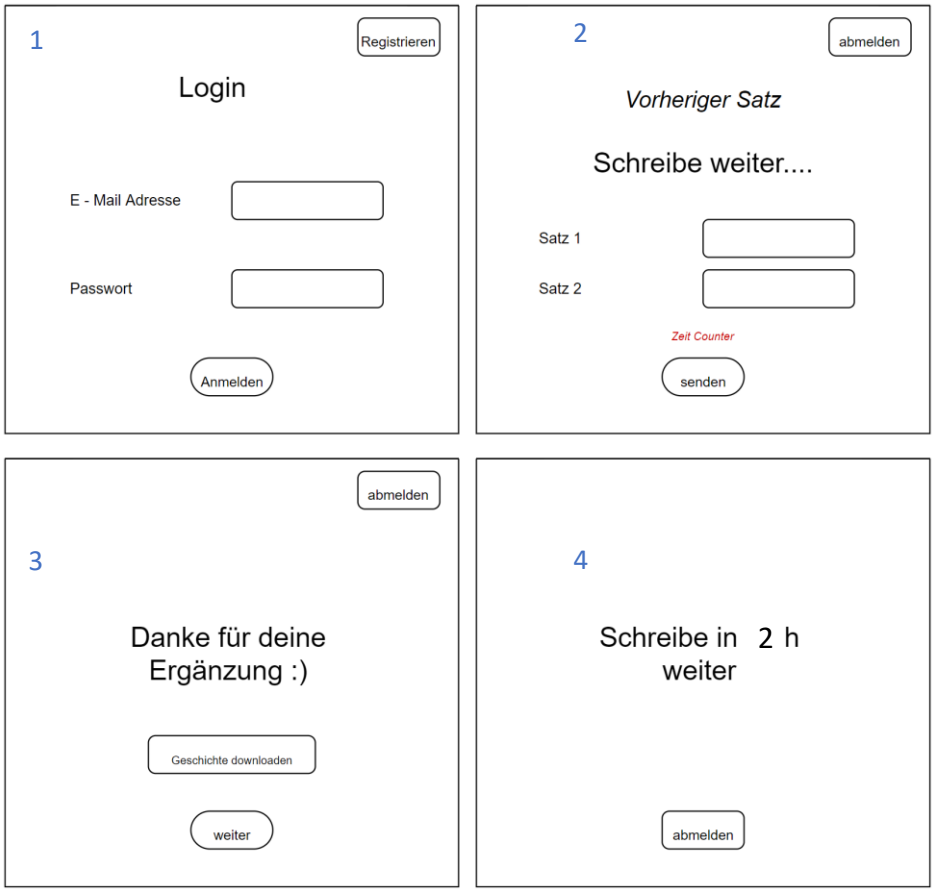


Abbildung 4 Wireframe

Falls man sich anmeldet, obwohl die 2h noch nicht vorbei sind, kommt folgendes Fenster, welches einem die verbleibende Wartezeit anzeigt. X steht in der Darstellung unten für die verbleibende Zeit. Das Registrierungsfenster ist gleich aufgebaut wie das Login Fenster.

Abbildung 5 Wireframe Zusatz

5.3.5 Testing

TC1 --> Login	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User gibt korrektes Passwort und E- Mail – Adresse ein. Login – Informationen werden überprüft. Login – Daten sind korrekt. User wird zur Hauptseite weitergeleitet.
Nachbedingung	keine
TC1.2 --> Login	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User gibt nicht korrektes Passwort und E- Mail – Adresse ein. Es kommt eine Information, dass die Eingabe nicht richtig war.
Nachbedingung	keine
TC2 --> Registrierung	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist noch nicht als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User drückt auf Registrieren – Button oben rechts. Es erscheint das Registrierung – Fenster. User gibt eine valide Eingabe der Email Adresse und des Passworts, gemäss definierter Werteberich ein und bestätigt seine Eingabe via. «registrieren»- Button. User wird in der DB gespeichert. User wird zur Hauptseite weitergeleitet.
Nachbedingung	keine
TC2.1 --> Registrierung	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist noch nicht als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User drückt auf Registrieren – Button oben rechts. Es erscheint das Registrierung – Fenster. User gibt eine

	invalide Eingabe der Email Adresse und des Passworts, gemäss definierter Werteberich ein und bestätigt seine Eingabe via. «registrieren»- Button. Es erscheint eine Information, dass die Registrierungsdaten nicht valide sind. Daten werden nicht in der Datenbank erfasst.
Nachbedingung	keine
TC3 --> Geschichte erweitern	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und befindet sich auf der Hauptseite
Beschreibung	User sieht den letzten Satz der Geschichte. Schreibt in die vorgesehenen Inputfelder zwei neue Sätze, gemäss definierten Wertebereich, aufgrund des letzten Satzes der Geschichte. Die Eingabe der neuen Sätze muss innerhalb zwei Minuten eingegeben werden. Die Eingabe wird mittels «senden»-Button bestätigt und in der Datenbank abgespeichert.
Nachbedingung	keine
TC3.1 --> Geschichte erweitern	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und befindet sich auf der Hauptseite
Beschreibung	User sieht den letzten Satz der Geschichte. Schreibt in die vorgesehenen Inputfelder zwei neue Sätze, nicht gemäss definierten Wertebereich, aufgrund des letzten Satzes der Geschichte. Es erscheint eine Information, dass die Eingabe nicht valide ist. Daten werden nicht in der Datenbank erfasst.
Nachbedingung	keine
TC4--> Geschichte herunterladen	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und hat seine zwei Sätze abgesendet.
Beschreibung	User wird nach dem Absenden der zwei Sätze auf die Herunterladen – Seite weitergeleitet. Durch den Download - Button hat der User die Möglichkeit die Geschichte herunterzuladen.
Nachbedingung	keine
TC5 --> Geschichte erweitern bevor 2h vorbei sind	

Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert und hat innerhalb der letzten zwei Stunden die Geschichte bereits erweitert.
Beschreibung	Bevor der User nach dem Anmelden auf die Hauptseite weitergeleitet wird, wird überprüft, ob er innerhalb der letzten zwei Stunden die Geschichte bereits erweitert hat. Falls dies der Fall ist, wird er nicht auf die Hauptseite weitergeleitet. Es erscheint eine Seite mit der Zeit Angabe, wie lange der User warten muss, bis er weiterschreiben darf.
Nachbedingung	keine

Abbildung 6 Testing Vorlage

5.4 Entscheiden

5.4.1 Versionsverwaltung

Für die Versionsverwaltung wird Git von GitHub gebraucht. Da ich dies für alle Projekte bei der Arbeit sowie für die Schule brauche bin ich mit Git bereits bekannt. Dieses Projekt wird jedoch nicht in der Siemens Umgebung umgesetzt sondern lokal, deshalb wird nicht das Siemensinterne Git verwendet sondern das kommerzielle GitHub.

Der Stand des Projekts wird mindestens nach jedem Tag auf Git abgelegt. Vorzugsweise wird die Arbeit auch nach jeder grösseren Änderung, zum Beispiel beim Erreichen eines Meilensteins, abgelegt.

GitHub Projektlink: https://github.com/laraaaf/M223.git

5.4.2 Projekt / Technologie

Folgende Mittel werden in diesem Projekt gebraucht:

- Programmiersprache: Java 11, SQL
- Datenbank: MySQL
- Betriebssystem: Windows 10
- Versionierungssystem: GitHub Repository und Git
- Diagramme (UML's): diagrams.net
- DIE: Visual Studio Code

Anders als im Projektauftrag beschrieben, wird Perl nicht verwendet. Dies ist so, da nicht viel im Backend verarbeitet werden muss. Es gibt lediglich einige Datenbank abfragen, welche gut direkt aus dem Java Frontend gemacht werden können. Falls die Daten zum Beispiel aus der DB oder vom Input mehr verarbeitet werden müssen, wäre ein effektives Backend mittels Perl vorzuziehen.

5.4.3 Design

Die Applikation berücksichtigt wegen mangelnder Zeit Barrierefreiheit nicht.

Navigation

Oben rechts im Fenster gibt es jeder Zeit die Möglichkeit, sich abzumelden. Beim Fenster Login kann man zum Registrieren wechseln. Bei der Registration zum Login. Die Navigation wird simpel gehalten. Die Buttons, mit welche man auf die nächste Seite kommt, befinden sich unten in der Mitte.

Geschichte

Die Geschichte wird beim Herunterladen in ein Textfile gespeichert Nach jeder Storyline (zwei Sätze eines Users) wird ein Umbruch gemacht. Falls ein Gamemaste die Geschichte herunterlädt, wird am Schluss einer Storyline der User angezeigt, welcher die Line verfasst hat. Die Geschichte wird im Downlaodsordner abgelegt.

Farb-/Schriftlegende

Schrift:

- Tahoma
- Times New Roman

Farbe:

- Schwarz
- Weiss
- Rot

Namenskonvention

- Variablen und Funktionen werden klein geschrieben (Camel Case)
- Files werden klein geschrieben

5.5 Realisieren

Das Projekt inkl. Dokumentation ist unter dem folgenden Link erreichbar: GIT - REPO

5.5.1 Persistenz

Es werden alle Informationen in der Datenbank gespeichert. Die Passwörter werden gehashed. Die Multiuser Funktionalität ist gegeben, da sich diverse User anmelden können und auf den gleichen Datenstand zugreifen.

5.5.2 Komponenten

Bei jedem Fenster ausser beim Login und Registration, gibt es die Möglichkeit sich abzumelden. Beim Fenster Login kann man zur Registration switchen und beim Fenster Registrieren zum Login Fenster.

5.5.2.1 Login

Die Login Seite dient als Entrypoint der Applikation. Um sich einzuloggen, muss man E-Mail Adresse sowie Passwort eingeben. Falls die Anmelde Informationen falsch sind, kommt ein Pop-UP Fenster mit der Information, dass Passwort oder E-Mail-Adresse falsch ist. Falls man sich anmeldet, und die letzte Erweiterung, die man gemacht hat, noch nicht 2h zurückliegt, kommt ein Fenster mit der Angabe, wie lange man noch warten muss.

```
//get id from user
String query = "SELECT ID_User FROM tbl_user WHERE Password = '" + password.hashCode() + "' AND Mail = '" + email.getText() + "'  ";
Statement sta = connection.createStatement();
ResultSet x = sta.executeQuery(query);

if (x.next() == false) {
    JOptionPane.showMessageDialog(anmeldenButton, "Passwort oder E-Mail Adresse Falsch");
} else {
    String id = x.getString(x.findColumn("ID_User"));

    //check if last expand was 2 hours ago
    query = "SELECT timestamp FROM tbl_userstory WHERE FK_User = '" + id + "' ORDER BY ID_StoryLine DESC";
    sta = connection.createStatement();
    ResultSet l = sta.executeQuery(query);
    if (l.next() == true){

    LocalDateTime startTime = LocalDateTime.parse(l.getString(l.findColumn("timestamp")),DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));

    LocalDateTime now = LocalDateTime.now();
    long difference = ChronoUnit.HOURS.between(startTime, now);

        You, seconds ago • Uncommitted changes
    if (difference>= 2){

        expandhistory frame = new expandhistory(id);
        frame.setVisible(true);
        dispose();
```

5.5.2.2 Registrierung

Falls man noch kein Login besitzt, kann man sich über das Registrierungsfenster registrieren. Man braucht eine E-Mail-Adresse gemäss dem definierten Wertebereich oben. Zudem wird ein Passwort gesetzt welches gehasht in die Datenbank gespeichert wird. Für das Hashen wird der Hash Algorithmus von der Java Library verwendet. Nach erfolgreichem registrieren, wird man zur Hauptseite (Geschichte erweitern) weitergeleitet.

```
//pattern
final String EMAIL_PATTERN =
"^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
+ "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";

final String passwort_pattern =
"([A-Za-z0-9\\?!,.-]+)";

//check if Mail is already used
String query = "SELECT Mail FROM tbl_user WHERE Mail = '" + email.getText() + "'  ";

Statement sta = connection.createStatement();
ResultSet res = sta.executeQuery(query);


if (!emailId.matches(EMAIL_PATTERN) || res.next() == true || !password.matches(passwort_pattern) || password.length() < 8 ) {
    JOptionPane.showMessageDialog(registrierButton, "E-Mail Adresse wird entweder bereits gebraucht oder Eingaben entsprechen nicht den vorgaben");
}else{


query = "INSERT INTO tbl_user (Password,Mail) values('" + hashpassword + "','" + emailId +  "')";
```

5.5.2.3 Geschichte erweitern

Es wird einem der letzte Satz der Geschichte angezeigt. Falls noch kein Satz in der Geschichte vorhanden ist, wird einem dies mitgeteilt und man schreibt die ersten zwei Sätze der Geschichte. Ansonsten erweitert man die Geschichte aufgrund des letzten Satzes um zwei neue. Seine Eingabe muss man mittels des Senden – Button abschicken. Diese werden in der Datenbank gespeichert. Zudem wird in der tab_userstory der Vorgang dokumentiert. Das heisst, es wird abgespeichert welcher User welche zwei Sätze erfasst hat. Nach dem Absenden wird der User auf die nächste Seite heruntergeladen.

```
String sent1 = s1.getText();
String sent2 = s2.getText();

//abspeichern der Sätze
String query = "INSERT INTO tbl_story (sentence1,sentence2) values('" + sent1 + "','" + sent2 +  "');
INSERT INTO `tbl_userstory` (`ID_StoryLine`, `timestamp`, `FK_User`, `FK_StoryLine`) VALUES (NULL, NOW(),'" + id + "',
(SELECT ID_StoryLine FROM `tbl_story` WHERE sentence1 = '" + sent1 + "' AND sentence2 = '" + sent2 + "') )";

Statement sta = connection.createStatement();
int x = sta.executeUpdate(query);
if (x == 0) {       You, a day ago • add Grundstruktur der Applikation
    JOptionPane.showMessageDialog(sendenButton, "konnte Geschichte nicht erweitern, verusche es erneut.");
} else {
    downloadhistory frame = new downloadhistory();
    frame.setVisible(true);
    dispose();
}
```

5.5.2.4 Geschichte herunterladen

Auf dieser Seite hat man die Möglichkeit, die bereits geschriebene Geschichte herunterzuladen. Diese wird in einem Text File in den Downloads Ordner abgelegt. Durch den Button «Weiter» gelangt man auf die nächste Seite. Falls der User ein Game-Master ist, wird ihm in der heruntergeladenen Geschichte angezeigt, welcher User was geschrieben hat.

```
//create output file
try {
    String home = System.getProperty("user.home");
    FileWriter myWriter = new FileWriter(home + "/Downloads/Geschichte.txt", Charset.forName("UTF8"));


    myWriter.write(x.getString("sentence1") + " ");
    myWriter.write(x.getString("sentence2") + "\n");

    while (x.next()){

            myWriter.write(x.getString("sentence1") + " ");
            myWriter.write(x.getString("sentence2") + "\n");

    }
    myWriter.close();
    JOptionPane.showMessageDialog(downloadButton, "Die Geschichte wurde heruntergeladen");

  } catch (IOException exception) {
    System.out.println("An error occurred.");
    exception.printStackTrace();
  }
```

5.5.2.5 Zeit Counter

Die zwei neuen Sätze müssen innerhalb von zwei Minuten erfasst und abgeschickt werden. Falls dies nicht geschieht, wird man wieder auf die Login Seite geleitet. Durch diese Funktion soll die Spontanität der Geschichte gefördert werden.

```
//counter
JLabel counter = new JLabel("Du hast nur 2 min!!");
counter.setFont(new Font("Tahoma", Font.PLAIN, 20));
counter.setForeground(Color.RED);
counter.setBounds(300, 400, 400, 50);
contentPane.add(counter);
You, an hour ago | 1 author (You)
Timer timer = new Timer(2*60*1000, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent arg0) {
        login frame = new login();
        frame.setVisible(true);
        dispose();

    }
});

timer.setRepeats(false); // Only execute once
timer.start();
```

5.6 Kontrollieren

Die Testings sind von den Use Cases abgeleitet. Durch das ist gewährleistet, dass alle Use Cases ebenfalls erfüllt sind. Pro Test Case gibt es unter Test Case, welche falsch Eingaben bzw. Grenz – Extremwerte berücksichtigen.

5.6.1 Testing

Die Tests wurden von Lara Felix auf dem Laptop mit dem, das Projekt umgesetzt wurde durchgeführt. Die Tests wurden am 07.12.2021 alle positiv getestet.

TC1 --> Login	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User gibt korrektes Passwort und E- Mail – Adresse ein. Login – Informationen werden überprüft. Login – Daten sind korrekt. User wird zur Hauptseite weitergeleitet. E-Mail: testing@user.ch Passwort: Passwort123!
Nachbedingung	keine
Resultat	Login – Informationen werden überprüft. Login – Daten sind korrekt. User wird zur Hauptseite weitergeleitet.
Status	OK
TC1.2 --> Login	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User gibt nicht korrektes Passwort und E- Mail – Adresse ein. Es kommt eine Information, dass die Eingabe nicht richtig war. E-Mail: testing@user.ch Passwort: Passwort12!
Nachbedingung	keine
Resultat	Es kommt eine Information, dass die Eingabe nicht richtig war.
Status	OK

TC2 --> Registrierung	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist noch nicht als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User drückt auf Registrieren – Button oben rechts. Es erscheint das Registrierung – Fenster. User gibt eine valide Eingabe der Email Adresse und des Passworts, gemäss definierter Werteberich ein und bestätigt seine Eingabe via. «registrieren»- Button. User wird in der DB gespeichert. User wird zur Hauptseite weitergeleitet. E-Mail: testing@[deinName].ch Passwort: Testing[deinName]!
Nachbedingung	keine
Resultat	User wird in der DB gespeichert. User wird zur Hauptseite weitergeleitet.
Status	OK
TC2.1 --> Registrierung	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist noch nicht als Benutzter registriert.
Beschreibung	Startet Applikation. Es erscheint das Login – Fenster. User drückt auf Registrieren – Button oben rechts. Es erscheint das Registrierung – Fenster. User gibt eine invalide Eingabe der Email Adresse und des Passworts, gemäss definierter Werteberich ein und bestätigt seine Eingabe via. «registrieren»- Button. Es erscheint eine Information, dass die Registrierungsdaten nicht valide sind. Daten werden nicht in der Datenbank erfasst. E-Mail: testing.ch Passwort: hallo
Nachbedingung	keine
Resultat	Es erscheint eine Information, dass die Registrierungsdaten nicht valide sind. Daten werden nicht in der Datenbank erfasst.
Status	OK
TC3 --> Geschichte erweitern	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und befindet sich auf der Hauptseite

Beschreibung	User sieht den letzten Satz der Geschichte. Schreibt in die vorgesehenen Inputfelder zwei neue Sätze, gemäss definierten Wertebereich, aufgrund des letzten Satzes der Geschichte. Die Eingabe der neuen Sätze muss innerhalb 2 Minuten eingegeben werden. Die Eingabe wird mittels «senden»-Button bestätigt und in der Datenbank abgespeichert. Satz1: Testing Satz. Satz2: Yuhui es funktioniert!
Nachbedingung	keine
Resultat	Die Eingabe wird mittels «senden»-Button bestätigt und in der Datenbank abgespeichert.
Status	OK
TC3.1 --> Geschichte erweitern	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und befindet sich auf der Hauptseite
Beschreibung	User sieht den letzten Satz der Geschichte. Schreibt in die vorgesehenen Inputfelder zwei neue Sätze, nicht gemäss definierten Wertebereich, aufgrund des letzten Satzes der Geschichte. Es erscheint eine Information, dass die Eingabe nicht valide ist. Daten werden nicht in der Datenbank erfasst. Satz1: hallo[] Satz2: halo ich bin ein viiiiiiiiiiiiel zu langer Satz.
Nachbedingung	keine
Resultat	Es erscheint eine Information, dass die Eingabe nicht valide ist. Daten werden nicht in der Datenbank erfasst.
Status	OK
TC3.2 --> Geschichte erweitern	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und befindet sich auf der Hauptseite
Beschreibung	User sieht den letzten Satz der Geschichte. Er schreibt uns sendet seine Sätze nicht innerhalb der vorgesehenen Zeit von 2 Min. Er wird nach Ablauf der zwei Minuten auf die Anmelde Seite weitregeleitet. Bereits eingegebenes wird nicht abgespeichert.

Nachbedingung	keine
Resultat	Es erscheint eine Information, dass die Eingabe nicht valide ist. Daten werden nicht in der Datenbank erfasst.
Status	OK
TC4--> Geschichte herunterladen	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert. User ist angemeldet und hat seine zwei Sätze abgesendet.
Beschreibung	User wird nach dem Absenden der zwei Sätze auf die Herunterladen – Seite weitergeleitet. Durch den Download - Button hat der User die Möglichkeit die Geschichte herunterzuladen.
Nachbedingung	keine
Resultat	Geschichte wird als Text File in den Downloadordner gespeichert.
Status	OK
TC5 --> Geschichte erweitern bevor 2h vorbei sind	
Bedingung	Applikation wurde betriebsbereit auf dem System installiert. User ist bereits als Benutzter registriert und hat innerhalb der letzten zwei Stunden die Geschichte bereits erweitert.
Beschreibung	Bevor der User nach dem Anmelden auf die Hauptseite weitergeleitet wird, wird überprüft, ob er innerhalb der letzten zwei Stunden die Geschichte bereits erweitert hat. Falls dies der Fall ist, wird er nicht auf die Hauptseite weitergeleitet. Es erscheint eine Seite mit der Zeit Angabe, wie lange der User warten muss, bis er weiterschreiben darf.
Nachbedingung	keine
Resultat	Es erscheint eine Seite mit der Zeit Angabe, wie lange der User warten muss, bis er weiterschreiben darf.
Status	OK

5.7 Auswerten

In diesem Teil werte ich meine Arbeit aus und formuliere hier mein Schlusswort.

5.7.1 Zeitplan (Soll/Ist Vergleich)

Mit dem Zeitplan ist es am Schluss gut aufgegangen. Am Anfang des Projektes war ich etwas hinterher, da ich unkonzentriert gearbeitet habe und viel geredet habe anstatt gearbeitet. Diesen Rückstand konnte ich jedoch beim Implementieren wieder aufholen.

5.7.2 Reflexion

Ich bin im Ganzen zufrieden mit meiner Arbeit. Es wurden alle Muss-Kriterien erfüllt und die Applikation wurde wie geplant umgesetzt. Ich denke, dass ich ein besseres Resultat hätte erreichen könne, hätte ich ihn den ersten 2-3 Tagen effizienter gearbeitet. Dies kann ihn einem nächsten Projekt erreichen, indem ich mich nicht mehr so schnell ablenken lasse. Wenn ich Musik höre und Geräuschunterdrückung einstelle, höre ich nicht mehr jedes Gespräch und rede nicht bei allem mit und kann den Fokus mehr auf die Arbeit legen. Dieses Vorgehen setzte ich, auch schon in der zweiten Woche also am Tag 4,5 und 6 um. So konnte ich die verlorene Zeit wieder aufholen. Mit ein wenig mehr Zeit hätte ich das Frontend- Design noch angepasst. Viele Labels bzw. Textboxen sind nicht korrekt zentriert. Zudem ist mein Code nicht gerade der schönste. Es funktioniert alles, jedoch hätte ich eines auslagern können, anstatt redundante Code – Schnipsel zu haben.

Das Erstellen der Applikation hat mir Spass gemacht. Ich arbeite zwar in meiner Abteilung mit Java, jedoch nicht oft und wenn schon, dann stehen schon diverse Template und Librarys zur Verfügung. Diese machen das Integrieren eines neuen Tools einfacher. Bei diesem Projekt musste ich jedoch alles vom Grunde auf neu erstellen. Zudem musste ich schon lange keine SQL-Abfragen mittels JOINS machen. Dieses veraltete Wissen konnte ich nun in diesem Projekt wieder auffrischen.

Zudem bekam ich in diesem Projekt einen Einblick wie die IPA verlaufen wird. Die Gespräche mit Remo, unserem HEX in diesem Projekt, haben mir immer sehr geholfen. Ich bekam Ratschläge, welche ich im Arbeitsjournal protokollierte und dann gleich umsetzte. Diese Tipps kann ich alle bei meiner IPA anwenden. Zudem habe ich mit diesem Dokument schon eine gute Vorlage für meine richtige IPA und weiss nun, wie die Struktur der IPA Dokumentation auszusehen hat.

5.8 Glossar

Wort	Bedeutung
Authentifizierung	Bestätigen, dass man ist, wer man ist. Z.B. mittels Identitätskarte oder Mail und Passwort
Autorisierungen	Definiert was ein User alles kann bzw. zu was er alles befugt ist
Client	zu Deutsch Kunden. Sind Personen, welche die Applikation benutzten.
DDL	Steht für Data Definition Language (DDL; Deutsch Datendefinitionssprache). Es ist eine Datenbanksprache, welche verwendet wird, um Datenstrukturen und verwandte Elemente zu beschreiben, zu ändern oder zu entfernen.
Hash Algorithmus	Funktion, um ein Stück Text oder anderes zu verschlüsseln
Multi-User-Applikation	Applikation, auf welche mehrere unabhängige User zugriffen könne. User haben verschiedene Autorisierungen und Authentifizierungen
UML's	Steht für Unified Modeling Language (vereinheitlichte Modellierungssprache). Es ist eine grafische Modellierungssprache von Software – Teilen und anderen Systemen.
Use Case	Use Cases dokumentieren die Funktionalität eines geplanten oder existierenden Systems auf Basis von einfachen Modellen
User	Steht für Benutzer einer Applikation
Wireframe	Ein Wireframe gibt das Design grob vereinfacht wieder und dient in erster Linie dazu, das Layout zu veranschaulichen.

5.9 Quellenverzeichnis

https://stackoverflow.com/

https://docs.oracle.com/en/java/

https://www.w3schools.com/

https://www.geeksforgeeks.org/

https://dev.mysql.com/doc/

5.10 Abbildungsverzeichnis

5.11 Tabellenverzeichnnis

6 Anhang

6.1 Login (Login.java)

```
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;


public class login extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField email;
```

```
    private JPasswordField passwordField;
    private JButton anmeldenButton;
    private JButton regButton;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    login frame = new login();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */

    public login() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(250, 190, 800, 700);
        setResizable(false);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
```

```
contentPane.setLayout(null);

JLabel lblNewUserRegister = new JLabel("Login");
lblNewUserRegister.setFont(new Font("Times New Roman", Font.PLAIN, 42));
lblNewUserRegister.setBounds(350, 52, 400, 50);
contentPane.add(lblNewUserRegister);

//email
JLabel lblNewLabel = new JLabel("E-Mail Adresse");
lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 20));
lblNewLabel.setBounds(242, 243, 160, 50);
contentPane.add(lblNewLabel);

email = new JTextField();
email.setFont(new Font("Tahoma", Font.PLAIN, 32));
email.setBounds(414, 235, 228, 50);
email.setColumns(10);
contentPane.add(email);


//passwort
JLabel lblPassword = new JLabel("Password");
lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 20));
lblPassword.setBounds(242, 324, 124, 36);
contentPane.add(lblPassword);

passwordField = new JPasswordField();
passwordField.setFont(new Font("Tahoma", Font.PLAIN, 32));
passwordField.setBounds(414, 320, 228, 50);
passwordField.setColumns(10);
contentPane.add(passwordField);
```

```
        //anmelden-button
        anmeldenButton = new JButton("Anmelden");
        anmeldenButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {



                String password = passwordField.getText();



                String msg = "" + email;
                msg += " \n";


                try {
                    Connection connection = DriverManager.getConnection("jdbc:mysql://localhost/foldingpaperstory", "root",
"");


                    //get id from user
                    String query = "SELECT ID_User FROM tbl_user WHERE Password = '" + password.hashCode() + "' AND Mail =
'" + email.getText() + "'  ";
                    Statement sta = connection.createStatement();
                    ResultSet x = sta.executeQuery(query);

                    if (x.next() == false) {
                        JOptionPane.showMessageDialog(anmeldenButton, "Passwort oder E-Mail Adresse Falsch");
                    } else {
                        String id = x.getString(x.findColumn("ID_User"));
```

```
                    //check if last expand was 2 hours ago
                    query = "SELECT timestamp FROM tbl_userstory WHERE FK_User = '" + id + "' ORDER BY ID_StoryLine
DESC";
                    sta = connection.createStatement();
                    ResultSet l = sta.executeQuery(query);
                    if (l.next() == true){

                    LocalDateTime startTime =
LocalDateTime.parse(l.getString(l.findColumn("timestamp")),DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));

                    LocalDateTime now = LocalDateTime.now();
                    long difference = ChronoUnit.HOURS.between(startTime, now);


                    if (difference>= 2){

                        expandhistory frame = new expandhistory(id);
                        frame.setVisible(true);
                        dispose();

                    }else{

                        long wait = 2 - difference;
                        time frame = new time(wait);
                        frame.setVisible(true);
                        dispose();
                    }
                }else{

                    expandhistory frame = new expandhistory(id);
```

```
                frame.setVisible(true);
                dispose();}

            }
            connection.close();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
});
anmeldenButton.setFont(new Font("Tahoma", Font.PLAIN, 22));
anmeldenButton.setBounds(320, 600, 150, 30);
contentPane.add(anmeldenButton);

//reg-Button
regButton = new JButton("Registrieren");
regButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {


        try {
            register frame = new register();
            frame.setVisible(true);
            dispose();

        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
});
regButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
```

```
        regButton.setBounds(600, 10, 150, 30);
        contentPane.add(regButton);

    }
}
```

6.2 Registration (register.java)

```
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;


public class register extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField email;
    private JPasswordField passwordField;
    private JButton registrierButton;
    private JButton anButton;

    /**
     * Launch the application.
     */
```

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                register frame = new register();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */

public register() {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(250, 190, 800, 700);
    setResizable(false);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblNewUserRegister = new JLabel("Registrierung");
    lblNewUserRegister.setFont(new Font("Times New Roman", Font.PLAIN, 42));
    lblNewUserRegister.setBounds(300, 52, 400, 50);
    contentPane.add(lblNewUserRegister);
```

```
//email
JLabel lblNewLabel = new JLabel("E-Mail Adresse");
lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 20));
lblNewLabel.setBounds(242, 243, 160, 50);
contentPane.add(lblNewLabel);

email = new JTextField();
email.setFont(new Font("Tahoma", Font.PLAIN, 32));
email.setBounds(414, 235, 228, 50);
email.setColumns(10);
contentPane.add(email);


//passwort
JLabel lblPassword = new JLabel("Password");
lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 20));
lblPassword.setBounds(242, 324, 124, 36);
contentPane.add(lblPassword);

passwordField = new JPasswordField();
passwordField.setFont(new Font("Tahoma", Font.PLAIN, 32));
passwordField.setBounds(414, 320, 228, 50);
passwordField.setColumns(10);
contentPane.add(passwordField);

//reg-button
registrierButton = new JButton("Registrieren");
registrierButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
```

```
                String emailId = email.getText();
                String password = passwordField.getText();
                int hashpassword = password.hashCode();

                try {

                    Connection connection = DriverManager.getConnection("jdbc:mysql://localhost/foldingpaperstory", "root",
"");


                 //Check if E-Mail input is valid

                    //pattern
                    final String EMAIL_PATTERN =
                    "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                    + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";

                    final String passwort_pattern =
                    "([A-Za-z0-9\\?!,.-]+)";

                    //check if Mail is already used
                    String query = "SELECT Mail FROM tbl_user WHERE Mail = '" + email.getText() + "'  ";

                    Statement sta = connection.createStatement();
                    ResultSet res = sta.executeQuery(query);


                    if (!emailId.matches(EMAIL_PATTERN) || res.next() == true || !password.matches(passwort_pattern) ||
password.length() < 8 ) {
                        JOptionPane.showMessageDialog(registrierButton, "E-Mail Adresse wird entweder bereits gebraucht
oder Eingaben entsprechen nicht den vorgaben");
```

```
                    }else{


                    query = "INSERT INTO tbl_user (Password,Mail,Rolle) values('" + hashpassword + "','" + emailId +  "',
0)";

                    sta = connection.createStatement();
                    int x = sta.executeUpdate(query);




                    if (x == 0) {
                        JOptionPane.showMessageDialog(registrierButton, "This is alredy exist");
                    } else {



                        //get id from user
                        query = "SELECT ID_User FROM tbl_user WHERE Password = '" + hashpassword + "' AND Mail = '" +
emailId + "'  ";
                        sta = connection.createStatement();
                        ResultSet ids = sta.executeQuery(query);
                        ids.next();
                        String id = ids.getString(ids.findColumn("ID_User"));


                        expandhistory frame = new expandhistory(id);
                        frame.setVisible(true);
                        dispose();
                    }
```

```
                connection.close();
                }
            } catch (Exception exception) {
                exception.printStackTrace();
            }
        }


});
registrierButton.setFont(new Font("Tahoma", Font.PLAIN, 22));
registrierButton.setBounds(320, 600, 150, 30);
contentPane.add(registrierButton);

//an-Button
anButton = new JButton("Anmelden");
anButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {


        try {
            login frame = new login();
            frame.setVisible(true);
            dispose();

        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
});
anButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
anButton.setBounds(600, 10, 150, 30);
```

```
        contentPane.add(anButton);

    }
}
```

6.3 Geschichte erweitern (expandhistory.java)

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;


import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
import javax.swing.Timer;


public class expandhistory extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField s1;
    private JTextField s2;
    private JButton sendenButton;
    private JButton abmeldenButton;

    /**
```

```
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    expandhistory frame = new expandhistory("1");
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */

    public expandhistory(String ID) {

        final String SentenceInput_Pattern =
        "([A-Za-z0-9\\?! ,.-]+)";


         String id = ID;
         try {
            Connection connection = DriverManager.getConnection("jdbc:mysql://localhost/foldingpaperstory", "root", "");


            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
setBounds(250, 190, 800, 700);
setResizable(false);
contentPane = new JPanel();
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
setContentPane(contentPane);
contentPane.setLayout(null);

//abfrage vorheriger Satz
String query = "SELECT sentence2 FROM `tbl_story` ORDER BY ID_StoryLine DESC";

Statement sta = connection.createStatement();
ResultSet x = sta.executeQuery(query);

if (x.next() == false) {
    JLabel lblsmattittle = new JLabel("Schreibe die ersätzen Sätze der Geschichte...");
    lblsmattittle.setFont(new Font("Times New Roman", Font.PLAIN, 30));
    lblsmattittle.setBounds(350, 52, 400, 50);
    contentPane.add(lblsmattittle);
} else {

    String s2 = x.getString(x.findColumn("sentence2"));
    JLabel lblsmattittle = new JLabel(s2);
    lblsmattittle.setFont(new Font("Times New Roman", Font.PLAIN, 30));
    lblsmattittle.setBounds(350, 52, 400, 50);
    contentPane.add(lblsmattittle);

}

JLabel lbltitle = new JLabel("Schreibe weiter...");
lbltitle.setFont(new Font("Times New Roman", Font.PLAIN, 42));
lbltitle.setBounds(300, 152, 400, 50);
```

```
contentPane.add(lbltitle);

//s1
JLabel s1label = new JLabel("Satz 1");
s1label.setFont(new Font("Tahoma", Font.PLAIN, 20));
s1label.setBounds(242, 243, 160, 50);
contentPane.add(s1label);


s1 = new JTextField(2);
s1.setFont(new Font("Tahoma", Font.PLAIN, 32));
s1.setBounds(414, 235, 228, 50);
s1.setColumns(10);
contentPane.add(s1);


//s2
JLabel s2label = new JLabel("Satz 2");
s2label.setFont(new Font("Tahoma", Font.PLAIN, 20));
s2label.setBounds(242, 324, 124, 36);
contentPane.add(s2label);

s2 = new JTextField();
s2.setFont(new Font("Tahoma", Font.PLAIN, 32));
s2.setBounds(414, 320, 228, 50);
s2.setColumns(10);
contentPane.add(s2);



//counter
```

```
    JLabel counter = new JLabel("Du hast nur 2 min!!");
    counter.setFont(new Font("Tahoma", Font.PLAIN, 20));
    counter.setForeground(Color.RED);
    counter.setBounds(300, 400, 400, 50);
    contentPane.add(counter);
    Timer timer = new Timer(2*60*1000, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            login frame = new login();
            frame.setVisible(true);
            dispose();

        }
      });

      timer.setRepeats(false); // Only execute once
      timer.start();

      System.out.println(timer.getDelay()/1000/60 + "min");




    //anmelden-button
    sendenButton = new JButton("senden");
} catch (Exception exception) {
    exception.printStackTrace();
    }
```

```
            sendenButton.addActionListener(new ActionListener() {
              public void actionPerformed(ActionEvent e) {

                if (!s2.getText().matches(SentenceInput_Pattern) || !s1.getText().matches(SentenceInput_Pattern) ||
s2.getText().length() > 25 || s1.getText().length() > 25 ){
                    JOptionPane.showMessageDialog(sendenButton, "Eingabe invalide");
                }else{



                 try {
                    Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost/foldingpaperstory?allowMultiQueries=true", "root", "");

                    String sent1 = s1.getText();
                    String sent2 = s2.getText();

                    //abspeichern der Sätze
                    String query = "INSERT INTO tbl_story (sentence1,sentence2) values('" + sent1 + "','" + sent2
+  "');INSERT INTO `tbl_userstory` (`ID_StoryLine`, `timestamp`, `FK_User`, `FK_StoryLine`) VALUES (NULL, NOW(),'" + id +
"', (SELECT ID_StoryLine FROM `tbl_story` WHERE sentence1 = '" + sent1 + "' AND sentence2 = '" + sent2 + "') )";

                    Statement sta = connection.createStatement();
                    int x = sta.executeUpdate(query);
                    if (x == 0) {
                        JOptionPane.showMessageDialog(sendenButton, "konnte Geschichte nicht erweitern, verusche es
erneut.");
                    } else {
                        downloadhistory frame = new downloadhistory(id);
                        frame.setVisible(true);
```

```
                dispose();
            }
            connection.close();
        } catch (Exception exception) {
            exception.printStackTrace();
            }
        }
    }
});

    sendenButton.setFont(new Font("Tahoma", Font.PLAIN, 22));
    sendenButton.setBounds(320, 600, 150, 30);
    contentPane.add(sendenButton);

    //reg-Button
    abmeldenButton = new JButton("abmelden");
    abmeldenButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {


            try {
                login frame = new login();
                frame.setVisible(true);
                dispose();

            } catch (Exception exception) {
                exception.printStackTrace();
            }
        }
    });
    abmeldenButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
```

```
            abmeldenButton.setBounds(600, 10, 150, 30);
            contentPane.add(abmeldenButton);


    }
}
```

6.4 Geschichte herunterladen (downloadhirstory.java)

```
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EmptyBorder;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.charset.Charset;


public class downloadhistory extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField s1;
    private JTextField s2;
    private JButton downloadButton;
    private JButton weiterButton;
    private JButton abmeldenButton;
```

```
/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                downloadhistory frame = new downloadhistory("0");
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */

public downloadhistory(String ID) {
    String id = ID;
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(250, 190, 800, 700);
    setResizable(false);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lbltitle = new JLabel("Danke \n für deine \n\n Ergänzung :)");
```

```
        lbltitle.setFont(new Font("Times New Roman", Font.PLAIN, 32));
        lbltitle.setBounds(200, 52, 400, 400);
        contentPane.add(lbltitle);


        //download-button
        downloadButton = new JButton("Geschichte downloaden");
        downloadButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {


                //Todo: herunterladen geschichte in txt

                try {
                    Connection connection = DriverManager.getConnection("jdbc:mysql://localhost/foldingpaperstory", "root",
"");

                    //check what rolle
                    String query = "SELECT Rolle FROM tbl_user WHERE ID_User = '" + id + "' ";
                    Statement sta = connection.createStatement();
                    ResultSet r = sta.executeQuery(query);
                    r.next();
                    String rolle = r.getString("Rolle");
                    System.out.println(rolle);

                    if (rolle.matches("0")){
                    query = "SELECT sentence1, sentence2 FROM tbl_story";}
                    else{ query = "SELECT `tbl_story`.`sentence1`, `tbl_story`.`sentence2`, `tbl_user`.`Mail` FROM
`tbl_story` LEFT JOIN `tbl_userstory` ON `tbl_userstory`.`FK_StoryLine` = `tbl_story`.`ID_StoryLine` LEFT JOIN `tbl_user`
ON `tbl_userstory`.`FK_User` = `tbl_user`.`ID_User`"; }
```

```
                    sta = connection.createStatement();
                    ResultSet x = sta.executeQuery(query);

                    if (x.next() == false) {
                        JOptionPane.showMessageDialog(downloadButton, "Die Geschichte hat noch keine Inhalt. Es wurden
keine Sätze gefunden");
                    } else {

                        //create output file
                        try {
                            String home = System.getProperty("user.home");
                            FileWriter myWriter = new FileWriter(home + "/Downloads/Geschichte.txt",
Charset.forName("UTF8"));

                            if (rolle.matches("0")){
                                myWriter.write(x.getString("sentence1") + " ");
                                myWriter.write(x.getString("sentence2") + "\n");

                                while (x.next()){

                                        myWriter.write(x.getString("sentence1") + " ");
                                        myWriter.write(x.getString("sentence2") + "\n");

                                }
                            }else{
                                myWriter.write(x.getString("sentence1") + " ");
                                myWriter.write(x.getString("sentence1") + " ");
                                myWriter.write(x.getString("Mail") + "\n");

                                while (x.next()){
```

```
                            myWriter.write(x.getString("sentence1") + " ");
                            myWriter.write(x.getString("sentence1") + " ");
                            myWriter.write(x.getString("Mail") + "\n");

                        }

                    }

                    myWriter.close();
                    JOptionPane.showMessageDialog(downloadButton, "Die Geschichte wurde heruntergeladen");

                } catch (IOException exception) {
                    System.out.println("An error occurred.");
                    exception.printStackTrace();
                }
            }
            connection.close();
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }
});
downloadButton.setFont(new Font("Tahoma", Font.PLAIN, 22));
downloadButton.setBounds(260, 450, 280, 30);
contentPane.add(downloadButton);

 //weiter-Button
 weiterButton = new JButton("weiter");
 weiterButton.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
```

```
            time frame = new time(2);
            frame.setVisible(true);
            dispose();

        }
    });
    weiterButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
    weiterButton.setBounds(320, 600, 150, 30);
    contentPane.add(weiterButton);

    //reg-Button
    abmeldenButton = new JButton("abmelden");
    abmeldenButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {


            try {
                register frame = new register();
                frame.setVisible(true);

            } catch (Exception exception) {
                exception.printStackTrace();
            }
        }
    });
    abmeldenButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
    abmeldenButton.setBounds(600, 10, 150, 30);
    contentPane.add(abmeldenButton);

}
```

```
}
```

6.5 Zeit Counter (time.java)

```
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;


import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;


public class time extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JButton weiterButton;
    private JButton abmeldenButton;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    time frame = new time(0);
                    frame.setVisible(true);
                } catch (Exception e) {
```

```
                e.printStackTrace();
            }
        }
    });
}

/**
 * Create the frame.
 */

public time(long Diff) {
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(250, 190, 800, 700);
    setResizable(false);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    //abfrage der Zeit
    JLabel lbltitle = new JLabel("Schreibe in " + Diff + " h weiter");
    lbltitle.setFont(new Font("Times New Roman", Font.PLAIN, 32));
    lbltitle.setBounds(200, 52, 400, 400);
    contentPane.add(lbltitle);


     //weiter-Button
     weiterButton = new JButton("fertig");
     weiterButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
```

```
                dispose();
            }
        });
        weiterButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
        weiterButton.setBounds(320, 600, 150, 30);
        contentPane.add(weiterButton);

        //reg-Button
        abmeldenButton = new JButton("abmelden");
        abmeldenButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {


                try {
                    register frame = new register();
                    frame.setVisible(true);
                    dispose();

                } catch (Exception exception) {
                    exception.printStackTrace();
                }
            }
        });
        abmeldenButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
        abmeldenButton.setBounds(600, 10, 150, 30);
        contentPane.add(abmeldenButton);

    }
}
```

6.6 Struktur Datenbank (db\ddl.sql)

```
/*

Author: Lara Felix

Titel: M223 - FoldingPaperStory

*/


-- DATABASE

DROP DATABASE IF EXISTS FoldingPaperStory;

CREATE DATABASE FoldingPaperStory;

USE FoldingPaperStory;


-- CREATE TABLES

CREATE TABLE tbl_User (

 ID_User INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,

 Mail VARCHAR(50),

 Password VARCHAR(50),

 Rolle Int

);


CREATE TABLE tbl_Story (
```

```
  ID_StoryLine INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  sentence1 VARCHAR(100),
  sentence2 VARCHAR(100)
);


CREATE TABLE tbl_UserStory (
  ID_StoryLine INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  timestamp DATE NOT NULL,
  FK_User INT UNSIGNED,
  FK_StoryLine INT UNSIGNED,
  FOREIGN KEY (FK_User) REFERENCES tbl_User(ID_User),
  FOREIGN KEY (FK_StoryLine) REFERENCES tbl_Story(ID_StoryLine)
);
```

6.7 Daten Datenbank (db\import.sql)

```
/*
Author: Lara Felix
Titel: M223 - FoldingPaperStory
*/


USE FoldingPaperStory;

DROP TABLE IF EXISTS tbl_user;

/*
writer
*/

INSERT INTO `tbl_user` (`ID_User`, `Mail`, `Password`, `Rolle`)
VALUES (
 NULL, 'testing@user.ch', '745064826', '0'
);
```

```
/*
gamemaster
*/

INSERT INTO `tbl_user` (`ID_User`, `Mail`, `Password`, `Rolle`)
VALUES (
 NULL, 'game@master.ch', '-1888845021', '1'
);
```