WS20/21 Softwaretechnik

Hausaufgabe 1: Zeitschätzung, Zeitplanung und Teamarbeit

Bei dieser Hausaufgabe arbeiten Sie in Teams an der Umsetzung eines Systems für die Zulassung zum neuen Informatik Studiengang. Dieses Projekt ist mit mehreren Herausforderungen verbunden, u.a. unzureichend spezifizierte Anforderungen, Zeitschätzungen und neue Technologien. Die Hausaufgabe ist so gestaltet, dass sie sich in erster Linie darauf konzentriert, praktische Erfahrungen in wichtigen nicht-technischen Aspekten der Arbeit an einem Software-Entwicklungsprojekt zu sammeln, wie z.B. Projektzeitabschätzung, Zeitplanung und Koordination innerhalb eines Teams. Daneben haben Sie auch die Möglichkeit, Ihre Fähigkeiten in der Softwareentwicklung zu verfeinern.

Die wichtigsten Lernziele dieser Hausaufgabe sind, dass Sie dazu in der Lage sind:

- Sorgfältige Zeitschätzungen vorzunehmen, sich Problemen dabei bewusst sein und Vermeidungsstrategien anzuwenden
- Projekte mit Aufgaben und Meilensteinen zu planen und zu terminieren und nach Bedarf umzuplanen
- Erste Risikobewertungen durchzuführen und Strategien zur Risikominderung zu planen
- Erste Entscheidungen über einen Prozess zu treffen und über Erfahrungen mit dem Prozess zu reflektieren
- Koordination zwischen den Teammitgliedern und Teamsitzungen effektiv durchzuführen.
- Über Erfahrungen mit der Arbeit in Teams zu reflektieren.
- In Entwicklungsprojekten mit Git erfolgreich zusammenzuarbeiten
- Sich für gute Software-Schreibpraktiken einzusetzen

Das Projekt

Die Universität zu Köln möchte in naher Zukunft einen Bachelor Studiengang „Informatik“ einführen. Zunächst sind für den Studiengang 200 Plätze vorgesehen, die Universität rechnet aber mit ca. 500 Interessierten pro Semester. Statt eines Numerus Clausus als Auswahlkriterium setzt die Universität auf ein Auswahlverfahren. Dazu will sie ein Softwaresystem zur Verwaltung des Zulassungsverfahrens für Informatik Studierende verwenden, einschließlich der Erfassung relevanter Informationen von den Bewerbern, der Annahme von Empfehlungsschreiben von Dritten, der Anzeige dieser Informationen beim Zulassungsausschuss und der Benachrichtigung der Bewerber*innen über die Entscheidung. Es ist allgemein bekannt, dass diese Art der Systeme aus unterschiedlichen Gründen von allen Beteiligten (Bewerber*innen, Unterstützer*innen und Zulassungsausschuss) als nicht besonders benutzerfreundlich angesehen werden. Immer wieder hört man auch von Fällen in denen es mit solchen Systemen zu peinlichen Pannen gekommen ist.[1]

Die Unileitung hat entschieden, dass sich die Universität zu Köln diese Blöße nicht geben will und hat Ihr Team damit beauftragt, ein neues Zulassungssystem für den neuen Informatik Studiengang zu entwickeln. Ihr Ansprechpartner an der Uni hat bisher nur gesagt, dass das neue

[1] http://www.cnn.com/2015/02/18/living/feat-carnegie-mellon-acceptance-letter-mistake/index.html

System "weniger Schrott" sei soll als andere vergleichbare Systeme. Als sie nach Details fragen bekommen Sie leiglich noch heraus, dass die Mitglieder des Zulassungsausschusses sich darüber beschweren, dass es in existierenden Systemen zu schwierig ist, einen Überblick über die Bewerber*innen zu bekommen, während die Bewerber*innen sich darüber beschweren, dass das System schwer zu bedienen ist. Darüber hinaus sei es für Schreiber von Empfehlungsschreiben schwierig, ihre Briefe mehrfach hochladen zu müssen, wenn sich Bewerber*innen auch noch für andere Studiengänge bewerben.

Es versteht sich natürlich von selbst, dass das neue System keine Zulassungsbescheide versenden sollte, wenn Studierende nicht angenommen werden.

Aufgaben

Ihr Team wird ein einfaches Zulassungssystem für Studieninteressierte einführen. Versuchen Sie das Optimum aus der knappen Zeit herauszuholen (Sie haben lediglich 3 Wochen Zeit). Sie werden wahrscheinlich nicht die Zeit haben, ein perfektes Produkt herzustellen, aber Sie sollten in der Lage sein, einen vernünftigen und brauchbaren Prototyp bereitzustellen. Planen Sie mindestens 50% Ihrer Zeit für nicht-coding Aktivitäten ein. Befolgen Sie die Teamrichtline aus dem Ilias Kurs (insbesondere die Sitzungsprotokolle, siehe unten).

Bei dieser Hausaufgabe haben Sie große Freiheit bei der Wahl der Methoden und Werkzeuge. Die einzigen expliziten Anforderungen sind:

1. Die gesamte Entwicklung erfolgt in einem von uns zur Verfügung gestellten GitHub-Repository. Halten Sie sich an gute Entwicklungspraktiken, (z.B. kohärente Commits mit sinnvollen Commit-Messages). Es ist nicht akzeptabel, wenn Sie Ihre Arbeit einfach am Ende mittels eines großen Commits in das Repository übertragen. Wir wollen sehen, dass Sie wirklich in dem Repository kollaborativ arbeiten.

2. Sie müssen ein Build-Automatisierungswerkzeug verwenden, um Abhängigkeiten zu verwalten und Ihr System zu bauen (z.B. Maven für Java).

3. Wenn Sie keine Implementierung zur Verfügung stellen, die ihr*e Betreuer*in *bequem* installieren und ausführen kann, müssen Sie mit ihm*ihr ein Termin für eine Demo ausmachen. Sprechen Sie mit Ihrem Betreuer*in ab, wie sie ihr System am einfachsten ausführbar machen können. Sie können beispielsweise ein Image einer virtuellen Maschine, ein Docker-Image oder eine vorkompilierte Datei zur Verfügung stellen.

Ihr Softwareprodukt kann als Client-Server-System mit einer Server- und einer Client-Anwendung, als webbasiertes System mit einem Browser als Client oder in jeder anderen Form strukturiert sein, die in diesem Szenario funktioniert. Für Java geben wir am Ende des Übungsblattes ein paar Tipps, wie Sie starten könnten, aber Ihr Team kann entscheiden, welche Programmiersprache und welcher Entwicklungsansatz am besten zu Ihren Bedürfnissen passt.

Prozessanforderungen

Wir haben nur wenige Anforderungen an den Entwicklungsprozess und lassen Ihnen ansonsten Freiheiten:

- Bevor Sie mit der Programmierung beginnen, sollten Sie
    - Alle Projektaufgaben planen. Sie sollten den Zeitbedarf für jede Aufgabe abschätzen und die Abhängigkeiten und Verantwortlichkeiten der Aufgaben dokumentieren.
    - Überlegen Sie sich, welchem Entwicklungsprozess Sie folgen wollen und welche Werkzeuge Sie verwenden werden. Zum Beispiel stellt GitHub einen Bugtracker zur Verfügung; Sie können die Verwendung von FindBugs als Teil Ihres Qualitätssicherungsprozesses in Betracht ziehen; erwägen Sie die Verwendung von Travis-CI; oder Sie können ein anderes Werkzeug verwenden.
    - Denken Sie über die mit diesem Projekt verbundenen Risiken und mögliche Dokumentationsstrategien nach.
- Verfolgen Sie während des gesamten Projekts Ihren Zeitaufwand, synchronisieren Sie die Aufgaben mit Ihrem Team und aktualisieren Sie den Plan regelmäßig. Verwenden Sie Sitzungsprotokolle, um zu verfolgen, wie Sie die Arbeit aufteilen.
- Erstellen Sie eine Dokumentation (max. 1 Seite) darüber, wie man Ihr System installieren kann und wie man es benutzt (als Endbenutzer). Das System sollte mit einem Build-Skript wie *make*, *Apache Ant*, *Apache Maven*, *Gradle* oder ähnlichem kompiliert werden. Denken Sie daran, bei Bedarf Abhängigkeiten in Ihr Projekt mit einzubeziehen.

Deadlines und Abgaben

Diese Hausaufgabe hat drei Deadlines und vier Ergebnisse. Zur ersten Deadline (Montag, 16. November, 23:59 Uhr) ist das Planungsdokument fällig. Der zweite Termin (Mittwoch, 25. November, 23:59 Uhr) gilt für die technischen Artefakte des entwickelten Systems. Die dritte Frist (Montag, 30. November, 23:59 Uhr) betrifft zwei Reflexionsdokumente. Wir haben diese Fristen absichtlich getrennt, um eine Vorabdokumentation des Planungsprozesses zu gewährleisten und Ihnen Zeit zu geben, über die Erfahrungen nachzudenken, ohne den Druck, gleichzeitig den Code für die Einreichung zu perfektionieren. Selbstverständlich können Sie (und sollten vielleicht auch) schon vor der zweiten Deadline mit dem Programmieren anfangen und vor der dritten Frist mit der Reflexion beginnen.

(0) Setup

Sie müssen ein GitHub-Repository für das Teamprojekt erstellen. Zuerst müssen Sie sich auf einen **Teamnamen** einigen. Ihr Teamname sollte eindeutig, aussprechbar, kurz (z.B. ein Wort) und etwas sein, das Sie in Ihrem Team mit Stolz auf den Straßen von Köln in Anwesenheit von Kindern anfeuern würden. Dann sollten Sie auf den folgenden Link gehen, um Ihr Team aufzustellen.

WARNUNG: Nachdem Sie einem Team beigetreten sind, können Sie das Team nicht mehr wechseln! Stellen Sie sicher, dass nur einer von Ihnen das Team zusammenstellt, und stellen Sie sicher, dass die übrigen Teamkolleg*innen dem richtigen Team beitreten.

Nachdem Sie die obige Warnung gelesen haben, gründen Sie ihr Team unter:

https://classroom.github.com/g/rNe3J6qg

(1) Planungsdokumente (fällig Montag, 16. November, 23:59) – 60 Punkte (24%)

- ***Ein erster Zeitplan.*** Sie können jedes Format wählen, solange der Inhalt klar rüberkommt (z.B. Netzpläne, Gantt-Diagramm, einfacher Text, Export aus einem Projektmanagement-Tool). Der Zeitplan sollte mindestens enthalten:
    - Aufgaben und Meilensteine,
    - Ergebnisse für alle Meilensteine,
    - Geschätzter Aufwand für alle Aufgaben,
    - Abhängigkeiten zwischen Aufgaben
    - Aufgabenzuweisungen für Teammitglieder (soweit bekannt)

  Sie sollten Ihre Zeiteinschätzung nachvollziehbar beschreiben (d.h. mit Erklärung, wie Sie zu diesem Wert gekommen sind). Die Aufgaben müssen in einer angemessenen Granularität dargestellt werden. Eine Daumenregel ist, dass jede Aufgabe nicht mehr als 1-2 Tage dauern sollte; wenn eine Aufgabe länger dauert, sollte sie in Teilaufgaben aufgeteilt werden.

- ***Eine einfache Risikobewertung.*** Identifizieren und beschreiben Sie auf max. einer Seite kurz die Hauptrisiken in diesem Projekt (mindestens zwei) und besprechen Sie die Strategien zur Risikominderung, die Sie anzuwenden gedenken.

- ***Ein erster Prozessplan.*** Beschreiben Sie auf max. einer Seite kurz den Prozess, den Sie zu befolgen gedenken. Mit Prozess meinen wir, wie Sie das System entwickeln werden und welche Schritte Sie befolgen werden, und nicht ein technisches Designmodell der Software. Wir sind insbesondere daran interessiert, wie Sie die Zusammenarbeit planen (z.B. Kommunikationskanäle, Sitzungen und deren Häufigkeit, Pull-Anfragen oder Dropbox) und welche Entwicklungsaktivitäten Sie insgesamt planen (z.B. wie viel Design, welche Qualitätssicherungsschritte, wie sollen Teile integriert werden).

Wir erwarten nicht, dass Ihr erster Plan aufgehen wird. Tatsächlich wären wir überrascht, wenn er es tut. Es ist wichtig, aus Misserfolgen zu lernen. Wir werden die Genauigkeit Ihrer Vorhersage nicht bewerten oder beurteilen, wie gut Sie an Ihrem ursprünglichen Plan festgehalten haben, sondern wir werden uns darauf konzentrieren, wie gut Sie Ihre Erfahrungen analysiert und reflektiert haben (siehe unten).

Das Ergebnis sollte als eine einzige PDF-Datei eingereicht werden, die in einem separaten Ordner in Ihrem Github Repository hochgeladen wird. Dieses Dokument sollte explizite Unterabschnitte für den Zeitplan, die Risikobewertung und den Prozessplan enthalten. Bitte geben Sie die Namen aller Teammitglieder auf der ersten Seite des Dokuments an zusammen mit einer Beschreibung, wer was zu welchen Teilen beigetragen hat. Mögliche Kategorien von Beiträgen sind z.B. Initiale Strukturierung des Kapitels, Text Produktion, Qualitätssicherung, Datenzulieferung, …

2) Code Artefakte (fällig, Mittwoch, 25. November, 23:59) – 50 Punkte (20 %)

Wir werden zum Stichtag einen Snapshot der Implementierung aus Ihrem GitHub-Repository erstellen. Ihr Repository sollte in der Datei README.md (oder von dort aus verlinkt) die Implementierung des Systems und eine kurze Dokumentation darüber enthalten, wie es gebaut/gestartet und wie es verwendet wird.

Halten Sie sich an gute Programmierpraktiken. Beispielsweise sollte Ihr Code eine klare Struktur haben, vernünftig modularisiert sein, geeignete Variablennamen verwenden und gegebenenfalls dokumentiert sein. Bonuspunkte gibt es, wenn das Repository zumindest eine minimale Form der Qualitätssicherung nachweist (z.B. FindBugs-Integration in Build-Prozess oder Testfälle). Wir werden uns die Commit-Historie ansehen; achten Sie daher bitte auf kohärente Commits mit aussagekräftigen Commit-Nachrichten.

Zusätzlich zum Quellcode sollten Sie, wenn möglich, ausführbare Images zur Verfügung stellen. Docker ist wahrscheinlich eine gute Lösung für die serverseitigen Komponenten, falls Sie welche haben. Clientseitige Komponenten können plattformspezifisch sein; Sie können ein VM-Image, ein Docker-Image oder eine Binärdatei zur Verfügung stellen. Auch hier gilt: Wenn Ihr*e Betreuer*in Ihre Binärdatei nicht ausführen kann, müssen Sie eine Besprechung anberaumen, um Ihr System zu demonstrieren. Der Link zu dem Image oder einer anderen Binärdatei sowie Login und Passwort, die in der virtuellen Maschine verwendet werden (falls zutreffend), sollten ebenfalls in der Datei README.md enthalten sein.

(3) Reflexionsdokuemnte – Team (fällig Montag, 30. November, 23:59) – 70 Punkte (28%)

Nachdem die Entwicklung abgeschlossen ist, reflektieren Sie über Ihre Erfahrungen. Tun Sie dies gemeinsam als Team. Auch hier legen wir Wert auf eine ehrliche Reflexion, die wahrscheinlich auch eine Reflexion über Misserfolge einschließen wird. Wir werden nicht bewerten, ob Sie den Aufwand richtig vorhergesagt haben, sondern vielmehr, was Sie aus dem Prozess gelernt haben. Das Reflexionsdokument soll aus 5 Teilen bestehen:

- **Tatsächlicher Zeitplan:** Dokumentieren Sie den tatsächlichen Zeitplan, einschließlich der Aufgaben, die Sie tatsächlich ausgeführt haben, und der tatsächlichen Zeit, die Sie dafür benötigt haben. Im Idealfall haben Sie Ihren Zeitplan während des gesamten Projekts auf dem neuesten Stand gehalten, was diese Aufgabe vereinfacht.
- **Abweichungen vom Zeitplan:** Denken Sie über die Unterschiede zwischen dem ursprünglichen und dem tatsächlichen Zeitplan nach. Welche Meilensteine wurden richtig vorhergesagt? Was wurde neu geplant? Gab es Aktivitäten, die Sie anfangs nicht geplant hatten oder die Sie am Ende fallen lassen mussten? Was waren die Gründe für die Änderungen? Hätten diese mit einer besseren Planung vorhergesehen werden können?
- **Prozess:** Denken Sie über den Prozess nach, den Sie ursprünglich geplant hatten, und über den Prozess, den Sie tatsächlich befolgt haben. War der Prozess angemessen? Haben Sie während des Projekts Schritte übersprungen oder zusätzliche Techniken angewandt? Mit welchen Herausforderungen sahen Sie sich konfrontiert? Wie könnte der Prozess verbessert werden, wenn Sie ein anderes, ähnliches Projekt durchführen müssten? Wie könnten Sie den Prozess ändern müssen, um sich an eine andere Art von Projekt anzupassen?
- **Erfahrung im Team:** Denken Sie über Ihre Erfahrung in der Arbeit im Team nach. Was hat gut funktioniert? Waren die Teamsitzungen effizient? Wie gut und über welche Prozesse haben Sie kommuniziert? Was muss verbessert werden? Gab es Herausforderungen bei der Teamarbeit, die Sie im Laufe des Projekts gelöst haben?

- **Sitzungsprotokolle:** Fügen Sie alle Sitzungsprotokolle bei, die während des gesamten Projekts geführt wurden und die Informationen über besprochene Themen, getroffene Entscheidungen und explizite Arbeitsaufträge enthalten sollten. Die Aufzeichnung der Ergebnisse von Besprechungen kann aufwendig sein; vielleicht sollten Sie diese Verantwortung unter den Mitgliedern Ihres Teams rotieren lassen.

Fassen Sie Ihre Ergebnisse zusammen und reichen Sie sie als eine einzige PDF-Datei mit expliziten Unterabschnitten ein. Laden Sie Ihre PDF-Datei in Ihrem Github Repository hoch. Die drei Reflexionsschritte Terminplanabweichungen, Prozess und Teamerfahrung sollten jeweils eine Seite nicht überschreiten. Es gibt keine Formatbeschränkungen für den Terminplan oder die Sitzungsprotokolle.

Einer der Hauptzwecke dieser Hausaufgabe besteht darin, eine eingehende Analyse der Gründe für eine gute oder schlechte Zeiteinschätzung, Terminplanung und Teamarbeitskoordination anzuregen. Schlechte Leistungen in diesen Bereichen sind nicht ungewöhnlich (wie zahlreiche Berichte aus realen Projekten zeigen). Deshalb werden wir nicht bewerten, wie gut (oder schlecht) das Projekt gelaufen ist, sondern vielmehr, wie gut Sie die Gründe verstanden haben, warum das Projekt so gelaufen ist, wie es gelaufen ist, und welche Lehren Sie aus Ihren Erfahrungen gezogen haben, die in Ihre zukünftige Arbeit einfließen sollen. Ein gutes Reflexionsdokument wird konkrete Aussagen über die gelernten Lektionen enthalten, mit klaren Belegen, wie z.B. Beispiele, um die Behauptungen zu untermauern. Es ist eine gute Idee, auf Ihre Sitzungsprotokolle zu verweisen, um Ihre Behauptungen zu untermauern und Beispiele zu liefern. Zum Beispiel: "Wir hätten besser kommunizieren können" wird nur schwach unterstützt. Man könnte es mit Beispielen aus der Entwicklungserfahrung wie folgt untermauern: "Eine Grund für [Mängeln/Entwicklungsverlangsamung/Qualitätsproblemen] war die Integration der Komponenten A und B, weil die API für A vom Entwickler von B nicht gut verstanden wurde… In Zukunft könnten wir versuchen, [diesen und jenen Prozess] zur klaren Dokumentation und Kommunikation solcher Design-Entscheidungen zu verwenden, anstatt [den Prozess, den wir befolgten/nicht befolgten].“

Die Fähigkeit, effektiv zu kommunizieren, ist eine wichtige Fähigkeit im Software-Engineering. Daher sollten Ihre Reflexionsdokumente gut geschrieben und leicht zu lesen sein. Stellen Sie sicher, dass Sie nach dem Schreiben Zeit für die Überarbeitung und das Korrekturlesen lassen. Es gibt viele praktische Tools für die gemeinschaftliche Textbearbeitung (Google Docs, Overleaf,…)

(4) Reflektionsdokument – Individuell (fällig, Montag, 30. November, 23:59) – 70 Punkte (28%)

(Beachten Sie, dass dies gleichzeitig mit dem Teamreflexionsdokument fällig ist).

Unabhängig von der Reflexion, die als Gruppe durchgeführt wird, sollte jedes Teammitglied individuell über (1) den Prozess, (2) die Zeitplanung und (3) die Teamarbeit nachdenken und diese Projekterfahrung mit seinen bisherigen Erfahrungen verbinden (vorzugsweise Erfahrungen aus einem nicht-akademischen Umfeld, aber Erfahrungen aus Lehrveranstaltungen sind auch möglich). Diese Reflexion sollte mindestens zwei der drei oben genannten Aspekte untersuchen. Sie müssen keine spezifischen Fragen beantworten, aber die folgenden Fragen

können mögliche Inhalte für die Reflexion anleiten: Wie deckt sich diese Projekterfahrung mit Ihren bisherigen Erfahrungen? Was war ähnlich? Was war anders? Was haben Sie persönlich aus dem Entwicklungsprozess dieses Projekts gelernt? Gibt es etwas, das Sie in Ihren zukünftigen Projekten anders machen wollen? Ähnlich wie bei der Teamreflexionsaufgabe werden wir die Qualität und Tiefe Ihrer Reflexion bewerten. Weitere Richtlinien finden Sie in den Anmerkungen zu den Hausaufgaben des Teams.

Reichen Sie das Dokument als einzelne PDF-Datei mit expliziten Unterabschnitten über Ilias ein. Das Gesamtdokument sollte 1,5 Seiten nicht überschreiten.

Anmerkungen

Diese Hausaufgabe (mit Ausnahme der individuellen Reflexion) ist in den Ihnen zugewiesenen Teams zu erledigen. Wir empfehlen Ihnen sehr, alle Fragen rund um das Team und Ihre Zusammenarbeit, offen zu diskutieren. Wenn schwerwiegende Probleme auftreten, wenden Sie sich früher an das Kurspersonal.

Benotung

In dieser Hausaufgabe werden 250 Punkte vergeben. Wir werden Sie auf der Grundlage der oben aufgeführten Lernziele benoten. Das ursprüngliche Planungsdokument umfasst 60 Punkte (24%), die technischen Artefakte und die Einhaltung des Prozesses 50 Punkte (20%), das Dokument zur Teamreflexion 70 Punkte (28%) und das Dokument zur Einzelreflexion 70 Punkte (28%).

Um die volle Punktzahl auf das Planungsdokument zu bekommen erwarten wir:

- Einen Zeitplan, der, wie oben beschrieben, Aufgaben, Meilensteine, Abhängigkeiten, etc. enthält.
- Eine Zeitschätzung für jede Aufgabe
- Eine Liste von mindestens zwei Risiken und entsprechenden Minderungsstrategien
- Ein Überblick über die Prozessschritte, die Sie in diesem Projekt verfolgen wollten.

Um die volle Punktzahl auf die technischen Artefakte zu bekommen erwarten wir:

- Eine kompilierende und laufende Prototyp-Implementierung
- Eine vorkonfigurierte Umgebung zur Ausführung Ihres Prototyps oder ein für eine Demo geplantes Treffen
- Angemessene Dokumentation für Systemadministratoren und Endbenutzer
- Angemessene Code-Struktur und -Stil, gegebenenfalls einschließlich Dokumentation
- Kohärente Commits von angemessener Größe mit aussagekräftigen Commit-Messages von allen Teammitgliedern
- Eine Form der grundlegenden Qualitätssicherung gibt Bonuspunkte

Um die volle Punktzahl auf die Reflexionsdokumente zu bekommen erwarten wir:

- Eine detaillierte, gut geschriebene und gut strukturierte Reflexion zu den oben aufgeführten Themen
- Ein Vergleich zwischen dem geplanten und dem tatsächlichen Zeitplan
- Eine Analyse, die über bloße Beschreibungen und oberflächliche Aussagen hinausgeht, einschließlich unterstützender Beweise für Behauptungen, die Aussagen über die Ursachen von Abweichungen, Konflikten usw. oder über Ihre eigenen Erfahrungen machen.
- Die Sitzungsprotokolle als Anlage

Ein paar Tipps, wenn Sie gar nicht wissen, wie Sie anfangen sollen

Oft ist der Start am schwierigsten: Wie war das nochmal mit der Konsole? Brauche ich eine IDE und wie konfiguriere ich die richtig? Wie bekomme ich denn mein Tool jetzt zum Laufen? Scheuen Sie sich nicht diese Fragen im Team anzusprechen und lassen Sie sich helfen oder helfen Sie anderen möglichst schnell eine gute Startbasis hinzubekommen. Davon wird ihr gesamtes Team profitieren.

Versuchen Sie nicht alles von Grund auf neu zu programmieren. Es gibt für fast alles gut dokumentierte Libraries und Frameworks, die Ihnen einen großen Teil der Arbeit abnehmen. Denken Sie aber auch nicht zu kompliziert. Für diese zeitlich sehr beschränkte Hausaufgabe werden Sie niemals den vollen Umfang der Frameworks brauchen.

Für diese Aufgabe bietet es sich z.B. an das System als eine einfache Web Applikation umzusetzen, die im Browser läuft. Für Java gibt es eine Reihe von Frameworks, die solche Anwendungen unterstützen:

- Spring Boot: https://spring.io/projects/spring-boot
- GWT (Google Web Toolkit): http://www.gwtproject.org/doc/latest/tutorial/index.html
- Grails: https://grails.org/

Mal ganz rudimentär heruntergebrochen braucht Ihre Anwendung wahrscheinlich irgendeine Form der persistenten Datenhaltung (in einer Datei, Datenbank, etc.), einen oder mehrere Controller, die die eigentliche Funktionalität steuern und umsetzen und eine Reihe von Ansichten, die die Daten und Formulare dann im Browser darstellen. Auch hierzu gibt es eine Reihe von Libraries, die das sehr einfach unterstützen und die zum großen Teil auch sehr gut mit den o.g. Web-Frameworks harmonieren. Überlegen Sie sich aber gut ob Sie eine simple Webseite nicht auch einfach schnell selber in html und ein wenig JavaScript anlegen (oder von einer Vorlage kopieren).

- JPA/Hibernate zur Datenpersistierung: https://hibernate.org/
- Spring MVC für das Web-Frontend: https://spring.io/guides/gs/serving-web-content/
- Angular für das Web-Frontend (basiert auf JavaScript, harmoniert aber auch mit einem Java Backend): https://www.baeldung.com/spring-boot-angular-web