SEP – Wintersemester 2023/2024

System zur Qualitätskontrolle in der Programmierausbildung

Lastenheft

Christian Bachmaier, Armin Größlinger, Stephan Lukasczyk

1 Motivation

Das Erstellen eines Computerprogrammes ist eine kreative Tätigkeit. Um gute Programme zu erstellen reicht es nicht aus, die Syntax einer Programmiersprache zu kennen; es braucht Erfahrung, Intelligenz, Geschmack und Geduld. Deshalb ist es in der Ausbildung im Bereich der Programmierung wichtig nicht nur theoretisches Wissen zu vermitteln, sondern im Mittelpunkt soll die praktische Umsetzung und das eigene Ausprobieren und Anwenden stehen. Es ist dabei wichtig größere Programme mit Praxisbezug zu schreiben, denn gängige Probleme treten bei kleinen und vereinfachten Beispielen erst gar nicht auf.

Um diesen Grundsätzen gerecht zu werden, soll ein webbasiertes System zur Qualitätskontrolle im Programmierpraktikum entwickelt werden. Ohne maschinelle Unterstützung ist es für Lehrende unmöglich, eine nicht-triviale Zahl von Studierenden/Praktikanten zu betreuen und eingereichte Programme/Code zu begutachten oder gar zu verbessern.

Das zu entwickelnde System soll zum Bewerten von Programmen bzw. deren Sourcecode verwendet werden können. Dabei soll es möglich sein, verschiedene Programmier-Aufgaben zu stellen, welche dann durch Studierende außerhalb des Systems (z. B. mit leistungsfähigen integrierten Entwicklungsumgebungen) gelöst und anschließend über das System zur Begutachtung eingereicht werden können. Desweiteren soll das System

eine Bewertung dieser Einreichungen ermöglichen (= *testieren*). Da in den meisten Fällen für die getätigten Einreichungen ein Leistungsnachweis, der ggf. sogar benotet ist, erstellt wird, ist ein absolutes Muss für so ein System, dass es robust und zuverlässig arbeitet. Ein prominentes und speziell Ihnen wohlbekanntes Beispiel ist Praktomat `https://praktomat.fim.uni-passau.de/praktomat/`.

2 Aufgabenstellung

Ziel des Software-Engineering Praktikums im Wintersemester 2023/2024 ist es, eine Webapplikation zu erstellen, welche innerhalb eines Kurses die zeitlich beschränkte Aufgabenstellung von Programmieraufgaben, die Entgegennahme von Lösungen, das automatische Blackbox-Testen von Lösungen und die Einsicht und abschließende manuelle Bewertung seitens einiger Tutoren ermöglicht.

Um die Anwendung einfach betreiben und verwalten zu können, sollte sie auf eigenem Webspace ausgeführt werden können. Die Anwendung sollte für einen Betreiber leicht einzurichten sein. Außerdem sollte dieser die Möglichkeit haben, ein gewisses Customizing der Anwendung vornehmen zu können, wie zum Beispiel das Festlegen eines Logos, des Namens des Betreibers bzw. der betreibenden Einrichtung, des Look & Feels, etc. Der oder die Betreiber fungieren als Administratoren des Systems.

Die zu erstellende Web-Anwendung soll intuitiv zu bedienen sein, geringe Anwendungshürden aufweisen, zugleich aber fortgeschrittenen Benutzern eine Vielzahl von erweiterten Funktionen mit einfacher Bedienung bieten.

Beispiele für ähnliche und bereits existierende Systeme sind Praktomat[1], ArTEMiS [2], Autograder.io[3], Autolab[4] und CodeGrade[5], um nur einige wenige Beispiele zu nennen. Oft gehen solche Systeme deutlich über die hier geforderte Funktionalitäten hinaus, z. B. sind sie selbst komplette Lernmanagementsysteme oder in solche eingebaut.

Die Anwendung muss mit Java und Jakarta Faces (JF) entwickelt werden. Auf Benutzerseite kann ein moderner Webbrowser vorausgesetzt werden.

3 Produkteinsatz

Zur Zielgruppe der zu entwickelnden Anwendung gehören als Betreiber Universitäten und Schulen mit eigenem Webspace. Der freie Zugang über das (freie) Internet soll für ein produktiv eingesetztes System möglich sein.

Ziel ist kein universelles Lernsystem wie Moodle oder Stud.IP, sondern speziell eines für Programmieraufgaben. Allerdings wäre denkbar, später in einem anderen Projekt, die Software mit einigen Features aus dieser Richtung zu erweitern.

[1] `https://praktomat.fim.uni-passau.de/`
[2] `https://github.com/ls1intum/Artemis`
[3] `https://eecs-autograder.github.io/autograder.io/`
[4] `https://autolabproject.com/`
[5] `https://www.codegrade.com/`

4 Produktfunktionen

4.1 Begriffsdefinitionen

Es gibt mehrere Gruppen von Benutzern (= Akteure) des Systems:

Administratoren Sind (meist als Universitätsmitarbeiter Betreiber der Plattform) für die Konfiguration des System verantwortlich. Sie können beispielsweise das Logo oder den Namen der Webapplikation festlegen, aber auch technische Einstellungen vornehmen. Wichtigste weitere Aufgaben eines Administrators sind die Verwaltung von Dozenten und die allgemeine Benutzerverwaltung. Ein Administrator hat jedoch allumfassende Rechte im System, z. B. auch an einzelnen Kursen.

Dozenten Dozenten werden von einem Administrator berechtigt, Kurse und darin Aufgaben inkl. Beschreibung, durchzuführende Checks und Zeitfristen anzulegen. Ein Dozent hat innerhalb eines eigenen Kurses Tutorrechte. Ein Dozent hat Einsicht und Änderungsrechte in alle Testate des Kurses. Außerdem hat er eine Übersicht über alle Bewertungen von Einreichungen aller Aufgaben des Kurses.

Tutoren Werden von Dozenten eines Kurses ernannt und können nach Ablauf der Abgabefrist Einreichungen zu Aufgaben des Kurses testieren, d. h. Durchsicht der Ergebnisse von automatischen Checks, Durchsicht des eingereichten Codes, mit Bemerkungen annotieren und bewerten.

Kursteilnehmer Sind registrierte Teilnehmer eines Kurses, die selbstständig die Aufgaben eines Kurses lösen und die entstandenen Lösungen am System einreichen. Diese werden automatisch durch installierte Checks geprüft. Kursteilnehmer können selbstverständlich nur ihre selbst eingereichten Lösungen und deren Testate einsehen.

Anonyme Benutzer Anonyme Nutzer können nur allgemeine Informationen über das System sehen, wie beispielsweise Kurzbeschreibung, Versionsnummer, eine Liste aller laufenden (Programmier-)Lehrveranstaltungen (= Kurse), etc. Zugriff auf weitere Informationen zu den Aufgabenstellungen ist ihnen *nicht* möglich. Falls Sie berechtigt sind, z. B. Kursteilnehmer, können Sie sich für einem Kurs registrieren. Eine Registrierung ist jeweils separat zu jedem Kurs notwendig. Hierfür ist eine Anmeldung am System erforderlich.

4.2 Minimale Anforderungen

Im Folgenden ist mit Erstellen, wenn nicht anders explizit erwähnt, gleichzeitig auch nachträgliches Editieren und Löschen gemeint.

Datenhaltung Das System soll für die Verwaltung von mehreren Veranstaltungen konzipiert sein wobei die Datenhaltung (Datenbank/Repositories) zwischen den einzelnen Veranstaltungen getrennt vorgenommen werden soll. Dies erleichtert datenschutzkonforme Sicherung und endgültiges Löschen von Daten nach Ablauf

der gesetzlichen Aufbewahrungsfristen von Prüfungsleistungen. Eine Veranstaltung (z. B. Programmierung II) gliedert sich jeweils in mehrere Aufgabenstellungen die von, zur Veranstaltung registrierten, Benutzern bearbeitet werden können.

Aufgabenerstellung Dozenten erstellen Aufgaben in eigenen Kursen im HTML Format inkl. möglicher Einbindung von Artefakten wie Bildern usw. Desweiteren geben sie Datum und Zeit an, ab wann eine Aufgabe für Kursteilnehmer veröffentlicht wird; damit ist die Aufgabe für die Teilnehmer sichtbar, einsehbar und kann bearbeitet werden. Dabei wird auch festgelegt, wann eine Aufgabe *empfohlen* und *endgültig* eingereicht werden muss. Hierbei wird unter *empfohlen* der im Ablauf der Veranstaltung als sinnvoll angesehene Abgabetermin und als *endgültig* der letztmögliche Termin festgelegt. Jede Aufgabenstellung hat eine Menge von *Checkern*, die beim Einreichen einer Lösung automatisch durchgeführt werden, jedoch auch manuell gestartet werden können.

Checker Checker sind Werkzeuge, die die Einreichungen nach bestimmten Kriterien überprüfen und dem Benutzer zurückmelden, ob die Überprüfung erfolgreich war oder nicht. Es können beliebig viele und verschiedene Checker beim Einreichen einer Lösung aufgerufen werden, um z. B. zu überprüfen, ob eine Datei hochgeladen werden konnte. Es soll ein Checker vorhanden sein, der den Code der eingereichten Lösung auf die maximale Zeilenbreite überprüft. Um eine Beeinflussung der Testierer zu vermeiden, soll in einem anderen Checker überprüft werden, ob sich Hinweise auf persönliche Daten im Code befinden. Ebenfalls soll überprüft werden können, ob eine eingereichte Lösung kompiliert oder nicht. Hierbei ist nur Java als Programmiersprache obligatorisch. Die Unterstützung anderer Programmiersprachen ist optional. Weitere denkbare Checker (z. B. funktionale Überprüfungen eines gestarteten Programms bzw. der einer Eingabe folgende Ausgaben eines Testfalls) sollen einfach und flexibel in das System integrierbar sein. Die Reihenfolge der Checkerausführung kann für einen sinnvollen Gesamttest, soweit sinnvoll, variiert werden. Als Optionen für einen Checker sind zumindest folgende Varianten vorzusehen:

- Voraussetzung zum Bestehen: Bei dieser Option kann bestimmt werden ob das Bestehen des Checks Voraussetzung für ein erfolgreiches Einreichen ist.
- Ergebnis einsehbar: Bei dieser Option kann bestimmt werden, ob die Ergebnisse eines Checks dem einreichenden Benutzer mitgeteilt werden.

Checker sollen auch manuell vom Testierer (nochmals) gestartet werden können, z. B. nachdem Checker verändert wurden.

Jeder Checker soll in einem eigenem Thread ausgeführt werden, da er potentiell lange laufen kann und Parallelität möglich wird. Während des Laufs der Checker wird der Benutzer kontinuierlich informiert (und die Browser-Session bleibt für den Ausführungszeitraum erhalten).

Einreichen Sobald eine Aufgabenstellung für Kursteilnehmer sichtbar wird, können Lösungen über ein Webformular eingereicht werden. Optional ist ein Einreichen

per Git-Push auf ein dafür bereitgestelltes Repository. Optional ist eine Editiermöglichkeit des Quellcodes im Webformular. Weitere Lösungen können bis zum Zeitpunkt, der für die endgültige Abgabe festgelegt wurde, eingereicht werden. Nach diesem Zeitpunkt steht die zuletzt eingereichte Lösung dem Testierer zum Testieren zur Verfügung. Nach einem Einreichungsversuch meldet das System, welche einsehbare Checker erfolgreich durchgelaufen sind („grün“ sind) bzw. welche nicht bestanden wurden („rot“ sind). Um eine Einreichung erfolgreich abzuschließen ist das Bestehen aller Checks die als *Voraussetzung zum Bestehen* markiert sind erforderlich. Erst nach dem erfolgreichen Einreichen wird die Lösung im System als solche zum Testieren gespeichert. Trotzdem werdem alle Inhalte eines Einreichungsversuch gezählt und aufbewahrt.

Unmittelbar nach jeder erfolgreichen Einreichung wird dem Einreicher eine Bestätigungs-E-Mail gesendet. Diese enthält zur Sicherheit auch den gesamten eingereichten Quellcode.

Eine Nebenbedingung stellt die Anonymität (persönliche Daten werden nicht angezeigt) der registrierten Benutzer dar, die über den gesamten Korrekturzyklus der Aufgabe bis zur endgültigen Bewertung gewahrt bleiben soll.

Testieren Wenn die Frist für die Einreichung abgelaufen ist (der Zeitpunkt der endgültigen Abgabe wurde überschritten) steht die jeweils letzte gültige Einreichung jedes Kursteilnehmers zum Testieren und Bewerten zur Verfügung. Testiert wird immer die letzte erfolgreiche Einreichung, aber ein Zugriff auf alle Einreichungen ist möglich, um diese z. B. systematisch auf die Evolution zur letzten Einreichung hin zu untersuchen. Dies kann auch außerhalb des Systems geschehen, z. B. nach Download einer Archivdatei. Auch der Testierer sieht unmittelbar die Anzahl der Einreichungsversuche. Den Testierern werden zu korrigierende Einreichungen jeweils zufällig zugewiesen. Die Ergebnisse aller bei der Einreichung durchgeführten Checker sollen dem Testierer übersichtlich dargestellt werden. Alle Checker können auch nochmals gestartet werden. Um eine Korrektur zu ermöglichen soll der Testierer im Webinterface folgende Möglichkeiten haben:

- Quellcode der Einreichung sichten und herunterzuladen um ihn z. B. in einer Enwicklungsumgebung zu debuggen.
- Kommentare zum Quellcode hinzufügen

Neben der kommentierten/korrigierten Fassung der Einreichung ist natürlich auch die Originalfassung im System zu erhalten. Zusätzlich soll der Testierer die Möglichkeit haben Kommentare zum Testat abzugeben. Hierbei können sowohl interne (nur für den Testierer ersichtlich) als auch für den einreichenden Nutzer ersichtliche Kommentare angegeben werden können.

Die Bewertung einer Einreichung soll jeweils in ganzen Noten von 1 bis 5 für die Kriterien *Verständlichkeit* und *Funktionalität*, wobei sich Verständlichkeit in *Layout* und *Struktur* unterteilt, erfolgen. Dabei bleibt, bis zum endgültigen Abschluss der Bewertung, der einreichende Benutzer gegenüber dem Testierer anonym. Die Note 6 soll für Täuschungsversuche vergeben werden können, wie z. B. Plagiate.

Lösungsdownload Um die Arbeit zu erleichtern soll es einem Dozenten möglich sein, ausgewählte (oder alle) Einreichungen einer Aufgabe als Archivdatei aus dem System herunter zu laden.

Testate einsehen Dozenten können immer alle Testate einsehen; Testierer können ihre selbst erstellten Testate einsehen und ändern. Kursteilnehmer können ihre Testate erst nach allgemeiner Freigabe aller Testate einer Aufgabe einsehen. Bei Freigabe werden Kursteilnehmer per E-Mail benachrichtigt, dass sie neue Testate auf dem System einsehen können. Es ist immer ersichtlich, welche Person ein Testat erstellt/zuletzt geändert hat.

Bewertungsübersicht Für den Dozenten steht immer eine Übersichtstabelle (= *Bewertungsstatistik*) der (bisherigen) Bewertungen aller Teilnehmer und Aufgabe eines Kurs zur Verfügung. Daraus kann er sich ein Bild für die Bewertung des Leistungsnachweises machen.

Registrierung Anonyme Personen können sich selbst zu einem Kurs registrieren. Dazu benötigen sie Namen, zugehörige Fakultäten und eine gültigen E-Mail-Adresse. Es muss verifiziert werden, dass die E-Mail-Adresse tatsächlich der registrierenden Person zugeordnet ist. Je nach Vorgabe in den Systemeinstellungen muss die E-Mail einem bestimmen regulären Ausdruck erfüllen, um z. B. Universitäts- oder Fakultätszugehörigkeit zu erzwingen.

Customizing Die Anwendung sollte die Möglichkeit bieten, die jeweilige betreibende Einrichtung zu präsentieren (Name, Logo, Farbschema, Kontaktmöglichkeiten, etc.) bzw. es an ein bereits etabliertes Look & Feel anzupassen. Auch Anwendungseinstellungen wie ein zwingendes E-Mail-Muster für Registrierungen kann vorgegeben werden.

Benutzerverwaltung Im System gibt es mehrere Rollen. Ein Admin kann alle Benutzer und die damit verbundenen Rollen verwalten: Anlegen neuer Benutzer, vollständiges Bearbeiten und Löschen existierender Benutzer und Zuordnen/Entziehen von Dozenten-, Tutor-, oder Adminrechten. Eine Funktion zum Suchen nach bestimmten Benutzern erleichtert für einen Admin die Verwaltung.

Login Ein Benutzer kann sich in ein Benutzerkonto einloggen und ist damit am System angemeldet. Hierfür ist eine Authentifizierung notwendig.

Kontoverwaltung Ein angemeldeter Benutzer kann sein Benutzerkonto editieren und eigenmächtig löschen.

Abmeldung Ein angemeldeter Benutzer kann sich vom System abmelden.

Online-Hilfe Der Benutzer muss zu jedem Zeitpunkt schnellen Zugriff auf die für die aktuelle Seite relevante Online-Hilfe haben.

5 Nicht-Funktionale Anforderungen

Folgend die Produktleistungen und Qualitätsanforderungen.

5.1 Usability

1. Einfache und intuitive Bedienbarkeit des Systems.
2. Die Benutzeroberfläche soll sich an allgemein geläufigen Bedienkonzepten und den damit verbunden Funktionen orientieren; alle Tabellen sollen z. B. nach den dargestellten Spalten sortierbar sein; Tabellen, die eine gewisse Größe überschreiten, sollen, um kurze Seitenladezeiten zu ermöglichen, auf mehrere Seiten aufgeteilt werden (Pagination).
3. Häufig wiederkehrende Aufgaben sind durch das System möglichst benutzerfreundlich zu unterstützen; häufig genutzte Funktionen sind möglichst einfach zugänglich zu machen.
4. Die Seiten der Applikation sind übersichtlich und einfach verständlich zu gestalten.
5. Daten sollen nicht nur leicht auffindbar und gut lesbar sein, sondern auch leicht einzugeben.
6. Bei Fehleingaben in ein Formular und der darauf folgenden Korrektur sollen die zuvor eingetragenen Felder nicht erneut einzugeben sein, sondern schon vorbesetzt sein. Außerdem sollte die Überprüfung der Eingaben nicht nach der ersten fehlerhaften Eingabe abbrechen, sondern alle Eingaben überprüfen und eine akkumulierte Fehlermeldung an den Benutzer zurückgeben.
7. Das System soll mit steigender Last skalieren.

5.2 Datensicherheit

1. Alle im System erfassten Daten sind persistent in einer Datenbank abzulegen; die Konsistenz der Daten ist sicherzustellen (Mehrbenutzerbetrieb!). Speziell, wenn Änderungen über mehrere Datenbanktabellen hinweg vorgenommen werden, sind Transaktionen zu nutzen.
2. Für die persistente Speicherung der Daten soll eine Datenbank (PostgreSQL) verwendet werden, die Ihnen von Ihren Betreuern zur Verfügung gestellt wird. Als Referenzplattform für die Implementierung dienen die Rechner im CIP-Pool.
3. Beim Löschen von Daten sind die Abhängigkeiten zwischen Daten einzelner Tabellen zu berücksichtigen; hat das Löschen eines Datensatzes das Löschen anderer Datensätze zur Folge, muss der Benutzer vorher deutlich darauf hingewiesen werden!

5.3 Datenschutz

1. Es muss sichergestellt werden, dass durch das System zu keinem Zeitpunkt sensible Daten für unberechtigte Dritte zugänglich sind.
2. Es sollen möglichst wenige technische Informationen über das System nach außen gegeben werden.
3. Alle personenbezogenen Daten, wie z. B. Login-Daten, sind sensibel und dürfen daher nur per SSL-Verbindung übertragen werden.
4. Passwörter dürfen nicht im Klartext gespeichert werden.
5. Die Nutzerdaten sind so zu speichern, dass kein unautorisierter Zugriff durch Dritte oder durch andere Nutzer im System stattfinden kann.
6. Das System darf nur im Rahmen der oben genannten oder vorgesehenen Funktionalitäten verändert werden.
7. Änderungen durch Manipulationen mit bekannten Angriffsmethoden wie SQL-Injection oder Cross-Site-Scripting müssen ausgeschlossen werden. Es müssen außerdem Maßnahmen ergriffen werden, um die Sitzungen der einzelnen Nutzer zu schützen (Session-Hijacking).

5.4 Internationalisierbarkeit

1. Für die Texte auf der Website ist die Zeichenkodierung UTF-8 zu wählen.
2. Die Sprache des Systems kann Deutsch oder Englisch sein; eine mehrsprachige Implementierung ist optional.

5.5 Evolutionsfähigkeit

Das zu erstellende System soll flexibel gegenüber zukünftigen Erweiterungen sein. Eine einfache und kostengünstige Weiterentwickelbarkeit des Systems ist sicherzustellen.

5.6 Installation

Es soll eine komfortable Installation für Systembetreiber bereitgestellt werden. Die Installation sollte einfach und schnell sein und automatisch das entsprechende Datenbank-Setup vornehmen.

6 Ergänzungen

Die Benutzung des Systems sollte mit allen gängigen Webbrowsern möglich sein. Wir raten allerdings davon ab, für jeden Browsertyp unterschiedlichen HTML-Code zu generieren. Der HTML-Code soll logisches Markup darstellen und nicht dazu missbraucht

werden, eine bestimmte graphische Darstellung zu erzwingen. Deswegen dürfen Features wie Frames nicht und explizites JavaScript nur nach Absprache eingesetzt werden. Der HTML-Code muss HTML-konform sein und z. B. durch den Validator des W3C[6] validierbar sein. Die Verwendung von Cascading Stylesheets (CSS) wird dringend angeraten.

Die Sessionverwaltung darf die Verwendung von Cookies nicht erzwingen.

Das System muss ein Log über alle Fehler führen, um das Debugging und den Betrieb der Anwendung zu vereinfachen. Achten Sie darauf, dass die Fehlerbeschreibungen detailliert genug sind, um auf einen Fehler bzw. dessen Ursache schließen zu können.

[6] `https://validator.w3.org/`