Antrag auf Festlegung einer Prüfungsordnung für Projektarbeiten im Fach Informatik.

Paul Jonas Dohle

25. November 2024

Inhaltsverzeichnis

1 Vorwort

Im digitalen Informationszeitalter ist ein breites Verständnis von Computersystemen und deren Funktionsweise von zentraler Bedeutung. Die Fähigkeit, eigene Software zu entwerfen und in einer Programmiersprache umzusetzen, ist dabei eine wesentliche Kompetenz, die SuS erwerben sollten. Dieses Wissen ermöglicht es ihnen, die logischen Prozesse nachzuvollziehen, die im Inneren eines Computers ablaufen, und ein tieferes Verständnis für die Technologien zu entwickeln, die unser Leben prägen.

Ein fundiertes Verständnis dieser Abläufe erleichtert nicht nur die verantwortungsvolle Nutzung digitaler Medien, sondern legt auch die Grundlage für eine kritische Auseinandersetzung mit aktuellen und zukünftigen technologischen Entwicklungen. Diese Kenntnisse fördern nicht nur die Medienkompetenz, sondern auch die Fähigkeit, technologiebezogene Entscheidungen bewusst und reflektiert zu treffen. In einer zunehmend digitalisierten Welt ist dies entscheidend, um die vielfältigen Möglichkeiten moderner Technologien sinnvoll zu nutzen, während gleichzeitig potenzielle Gefahren erkannt und minimiert werden können. Letztlich trägt dies dazu bei, die SuS zu selbstbewussten, verantwortungsvollen und kritisch denkenden Teilnehmern der digitalen Gesellschaft zu machen.

Das Wahlpflichtfach Informatik der Klassenstufe 9 und 10 spielt eine entscheidende Rolle, um SuS das notwendige Wissen und die Kompetenzen zu vermitteln, die sie für die selbstbestimmte Teilhabe in der digitalen Welt benötigen. Hier werden die Grundlagen geschaffen, um Computersysteme nicht nur zu verstehen, sondern auch aktiv mit ihnen zu arbeiten. Im Unterricht sollen die SuS lernen, wie digitale Technologien funktionieren, und erlangen Einblicke in Themen wie Algorithmen, Datenstrukturen, Netzwerke und Programmierung. Diese Fähigkeiten sind nicht nur für eine mögliche berufliche Zukunft im Bereich der IT relevant, sondern fördern auch allgemein das logische Denken und die Problemlösungsfähigkeiten.

Ein zentrales Ziel des Informatikunterrichts ist es, den SuS Werkzeuge an die Hand zu geben, mit denen sie digitale Prozesse analysieren, hinterfragen und gestalten können. Dies umfasst das Verständnis von Automatisierung und Datenverarbeitung genauso wie den sicheren Umgang mit Technologien im Alltag. Indem sie selbst Programme schreiben und Systeme entwickeln, erleben die SuS, wie sie eigene Ideen in digitale Lösungen umsetzen können. Dies stärkt ihr technisches Selbstbewusstsein und zeigt, dass sie nicht nur Konsumenten, sondern auch aktive Gestalter der digitalen Welt sein können.

Darüber hinaus leistet der Informatikunterricht einen wichtigen Beitrag zur Förderung der Medienkompetenz. Themen wie Datenschutz, Cyber-Sicherheit

und logische Denkprozesse sind feste Bestandteile des Unterrichts. Die SuS lernen, kritisch mit Informationen umzugehen, digitale Inhalte zu bewerten und ihre Privatsphäre zu schützen. Dies befähigt sie, nicht nur technische Probleme zu lösen, sondern auch gesellschaftliche Herausforderungen im Kontext der Digitalisierung zu erkennen und zu bewältigen.

2 Die Problematik der Unklarheit der Bewertungsrichtlinien

Im Lehrplan des Europa-Gymnasiums Warstein ist im Fach Informatik in der Klassenstufe 10 eine Unterrichtseinheit zur Programmiersprache Python angesetzt. Dort sollen die SuS an ihre Kenntnisse des vorhergehenden Unterrichtsvorhabens anknüpfen und die spielerisch erlernten Konzepte durch textuelle Programmierung vertiefen. Zum Abschluss dieser Einheit sieht der Lehrplan eine fakultative Projektarbeit vor. Hier können die SuS Ihre Kreativität entfalten und die erlernten Fähigkeiten unter Beweis stellen. Im Gegensatz zu isolierten Beispielaufgaben, kann eine Projektarbeit eine große Bandbreite von Konzepten und Prinzipien abdecken.

In der Kürze von lediglich 30 Unterrichtsstunden kann auf viele Dinge selbstverständlich nur oberflächlich eingegangen werden. Besitzen einzelne SuS bereits Vorkenntnisse im genannten Bereich, so lassen sich diese Wissensdifferenzen nicht im besagten Zeitraum überwinden. Somit sind große Disparitäten zwischen Arbeiten einzelner Schüler erwartbar und müssen als solche hingenommen werden.

2.1 Funktion von Projektarbeiten

Basierend auf den Vorstellungen der Lehrkraft und den Rahmenbedingungen des Lehrplans kann die Projektarbeit unterschiedliche Zielsetzungen verfolgen. Wird sie als Äquivalent zu einer Klassenarbeit konzipiert, liegt der Fokus primär auf der bloßen Wissensabfrage. In diesem Fall dient die Projektarbeit dazu, die Fähigkeit der SuS zu prüfen, die im Unterricht vermittelten Inhalte und Befehle korrekt anzuwenden. Die Bewertung konzentriert sich hierbei ausschließlich auf die syntaktische Korrektheit und die Funktionsfähigkeit des erstellten Projektcodes.

Andere Aspekte wie Lesbarkeit des Codes, Qualität der Dokumentation oder die Effizienz der Lösungsansätze bleiben in dieser Betrachtung außen vor. Stattdessen steht im Vordergrund, ob die SuS die zentralen Konzepte der Programmiersprache verstehen und technisch korrekt umsetzen können. Diese Form der Bewertung ermöglicht eine klare und objektive Einschätzung der erlernten Grundlagen und stellt sicher, dass die SuS mit den wichtigsten Bausteinen der Programmierung vertraut sind.

Eine alternative Herangehensweise besteht darin, die Projektarbeit als Abbildung des gesamten Softwareentwicklungsprozesses zu gestalten. Dabei

wird den SuS bewusst ein großer Gestaltungsspielraum eingeräumt, um realitätsnahe und kreative Ergebnisse zu fördern. Der Fokus liegt nicht nur auf der technischen Umsetzung, sondern auf einem ganzheitlichen Verständnis der Softwareentwicklung, das auch unabhängig von einer spezifischen Programmiersprache anwendbar ist.

In diesem Rahmen geht es nicht nur darum, die Syntax zu beherrschen, sondern vor allem darum, die Fähigkeit zu entwickeln, eigenständig und methodisch Software zu entwerfen und umzusetzen. Von den SuS wird erwartet, dass sie ein geplantes und strukturiertes Vorgehen zeigen, das typische Schritte der Softwareentwicklung wie Anforderungsanalyse, Konzeption, Implementierung und Testen umfasst.

Darüber hinaus fließen in die Bewertung auch qualitative Aspekte der Arbeit ein. Die Lesbarkeit und Wartbarkeit des Codes, die Qualität der Dokumentation sowie die Überlegungen zur Skalierbarkeit oder Erweiterbarkeit der Software spielen hier eine Rolle. Diese Kriterien orientieren sich an den Anforderungen der realen Softwareentwicklung und bereiten die SuS darauf vor, auch komplexere Projekte strukturiert und nachhaltig umzusetzen.

Eine solche Ausrichtung der Projektarbeit zielt darauf ab, nicht nur die technischen Fertigkeiten der SuS zu fördern, sondern ihnen auch ein Verständnis dafür zu vermitteln, welche Bedeutung eine sorgfältige Planung und eine durchdachte Umsetzung für den Erfolg eines Softwareprojekts haben. Sie lernen, dass gute Software nicht nur funktioniert, sondern auch langfristig wartbar, verständlich und anpassbar sein sollte.

Durch diese Herangehensweise werden nicht nur grundlegende Programmierkenntnisse vertieft, sondern auch übergreifende Kompetenzen vermittelt, die die SuS auf weiterführende Herausforderungen in Studium, Beruf oder privaten Projekten vorbereiten.

2.2 Unklarheit der Bewertungsrichtlinien

Die genaue Funktion der Projektarbeit beeinflusst unmittelbar die Rahmenbedingungen, wie die zulässigen Hilfsmittel und die festgelegten Bewertungskriterien. Um sicherzustellen, dass alle SuS die gleichen Voraussetzungen haben und eine faire Vergleichbarkeit gewährleistet ist, ist es unerlässlich, diese Regelungen vor Beginn der Projektphase klar und verbindlich zu kommunizieren. Ein Mangel an solchen Vorgaben führt unweigerlich zu Missverständnissen und Konflikten, insbesondere wenn unterschiedliche Projektgruppen das Ziel oder die Anforderungen der Arbeit unterschiedlich interpretieren.

Ohne klare Richtlinien wird auch eine faire Bewertung erheblich erschwert, da divergierende Ansätze der SuS nicht objektiv miteinander verglichen wer-

den können. Eine transparente und einheitliche Festlegung der Rahmenbedingungen hilft, Missverständnisse zu vermeiden und schafft eine Grundlage für eine gerechte Beurteilung der Arbeit. Zu den Regelungen, die vorab definiert werden sollten, gehören unter anderem:

- **Verwendung von Fremdcode:** Es sollte klar festgelegt werden, ob und in welchem Umfang die Nutzung von fremdem Quellcode erlaubt ist und wie dieser gekennzeichnet werden muss, um Plagiate zu vermeiden.
- **Einsatz von KI:** Da Tools wie ChatGPT oder andere generative KI-Modelle immer mehr genutzt werden, ist es wichtig zu definieren, ob sie als Hilfsmittel zugelassen sind und wie deren Verwendung dokumentiert werden muss.
- **Bewertungskriterien:** Neben der Funktionsfähigkeit der Software sollten weitere Aspekte wie Code-Sauberkeit, Dokumentation, Testabdeckung, Wartbarkeit und Modularität berücksichtigt werden. Diese Kriterien müssen klar und für alle verständlich formuliert sein, um subjektive Bewertungen zu minimieren.

Durch die frühzeitige Klärung dieser Punkte werden nicht nur Konflikte vermieden, sondern auch eine Grundlage für ein produktives Arbeiten und eine faire Beurteilung geschaffen. Darüber hinaus können die SuS durch klare Vorgaben effektiver arbeiten und gezielt Kompetenzen entwickeln, die über das Projekt hinaus von Bedeutung sind.

3 Mögliche Lösungsansätze

Unter Berücksichtigung aller relevanten Faktoren ist es unwahrscheinlich, eine ideale Lösung zu finden, die alle Herausforderungen und Bedürfnisse gleichermaßen abdeckt. Dennoch gibt es verschiedene Ansätze, um die beschriebene Problematik möglichst effektiv zu minimieren und eine praktikable Lösung zu erreichen. Durch sorgfältige Planung, klare Kommunikation und gezielte Anpassungen der Rahmenbedingungen kann eine Balance zwischen den Anforderungen und den realistischen Gegebenheiten geschaffen werden.

3.1 Reine Syntaxabfrage

Die SuS sollen nachweisen, dass sie die im Unterricht behandelten Befehle der Programmiersprache korrekt anwenden können. Der Schwerpunkt liegt auf der Beherrschung der Syntax, nicht auf der Fähigkeit, eigenständig komplexe Software zu entwickeln. Ziel ist es, zu zeigen, dass die SuS die grundlegenden syntaktischen Strukturen der Programmiersprache verstehen und fehlerfrei umsetzen können. Ein tiefergehendes Verständnis der zugrunde liegenden Konzepte oder eine eigenständige Entwicklung von Softwarelösungen ist in diesem Kontext nicht erforderlich.

§ 1 Die SuS müssen den gesamten Code selbst entwickeln.

§ 2 Der Einsatz von KI ist grundsätzlich untersagt.

§ 3 Die Verwendung von Fremdcode, der nicht von der SuS stammt, ist untersagt.

§ 4 Die einzige zulässige Hilfestellung eines Code-Editors ist Syntax-Highlighting. Vorschläge, Debugger und sonstige Erweiterungen sind nicht erlaubt. Die Nutzung der IDLE wird empfohlen.

§ 5 Die SuS dürfen keine externen Bibliotheken oder Frameworks verwenden, außer es wird explizit erlaubt. Alle Lösungen müssen mit den im Unterricht behandelten Mitteln erstellt werden.

§ 6 Tests, Kommentare, Code-Optimierungen oder zusätzliche Features, die über die gestellten Anforderungen hinausgehen, werden nicht berücksichtigt. Die Bewertung erfolgt ausschließlich auf Basis der syntaktischen Korrektheit und Lauffähigkeit des Codes.

3.2 Grundlegende Code-Funktionalität

Die SuS müssen die grundlegende Funktionalität des Programms selbstständig entwickeln. Der Fokus liegt auf der korrekten Umsetzung der Anforderungen, wobei die Funktionalität im Vordergrund steht. Die SuS dürfen Hilfsmittel wie externe Bibliotheken verwenden, müssen jedoch in der Lage sein, die erzeugten Lösungen zu verstehen und die angewendete Logik zu erklären. Die Lösung muss die gestellten Anforderungen erfüllen und in einer sauberen, syntaktisch korrekten Weise implementiert sein. Testmodule und zusätzliche Features, die über die Hauptanforderungen hinausgehen, sind nicht erforderlich und werden nicht bewertet.

§ 1 Die SuS müssen den des Codes größtenteils selbstständig entwickeln.

§ 2 Der Einsatz von KI ist grundsätzlich untersagt.

§ 3 Die Verwendung von Fremdcode, der nicht von der SuS stammt, ist lediglich zu Lern-/ Recherchezwecken gestattet.

§ 4 Die einzige zulässige Hilfestellung eines Code-Editors ist Syntax-Highlighting. Vorschläge, Debugger und sonstige Erweiterungen sind nicht erlaubt. Die Nutzung der IDLE wird empfohlen.

§ 5 Die SuS müssen in der Lage sein, den verwendeten Code zu erklären, einschließlich der angewendeten Algorithmen und Strukturen.

§ 6 Grundsätzlich wird nur die syntaktische Korrektheit gewertet. In Extremfällen kann auch die Lesbarkeit in die Bewertung einfließen.

§ 7 Die Bewertung, ob die Kriterien im Einzelfall eingehalten wurden, behält sich die Lehrkraft vor.

3.3 Fokus auf Codequalität und Struktur

Die SuS sollen hauptsächlich funktionalen Code entwickeln, jedoch auch auf grundlegende Aspekte der Lesbarkeit und Verständlichkeit achten. Der Code sollte klar strukturiert und gut organisiert sein, mit sinnvollen Variablennamen und einer modularen Struktur. Einfache Hilfsmittel wie ein Code-Editor mit Vorschlägen und integriertem Debugger sind gestattet, jedoch muss der Code so geschrieben werden, dass er auch ohne diese Tools gut lesbar und wartbar bleibt. Zur Lösung unbekannter Probleme ist auch die Verwendung von Fremdcode zulässig, solange die SuS die verwendeten Code-Schnipsel verstehen und in der Lage sind, deren Funktion zu erklären.

§ 1 Die SuS müssen den des Codes großteils selbstständig entwickeln.

§ 2 Der Einsatz von KI ist grundsätzlich untersagt.

§ 3 Die Verwendung von Fremdcode, der nicht von der SuS stammt, ist in Einzelfällen gestattet.

§ 4 Der Code Editor darf bei grundlegenden Aufgaben durch Vorschläge und Warnungen unterstützen. Die Verwendung eines Debuggers und weitere fundamentaler Hilfsmittel sind nach Absprache ebenfalls zulässig.

§ 5 Neben der syntaktischen Korrektheit werden auch grundlegende Aspekte der Lesbarkeit gewertet.

§ 6 Die SuS müssen in der Lage sein, den verwendeten Code zu erklären, einschließlich der angewendeten Algorithmen und Strukturen.

§ 7 Die Bewertung, ob die Kriterien im Einzelfall eingehalten wurden, behält sich die Lehrkraft vor.

3.4 Erweiterter Kenntnisnachweis

Im Rahmen des erweiterten Kenntnisnachweises wird von den SuS erwartet, dass sie nicht nur funktionalen Code entwickeln, sondern auch ein gutes Verständnis der zugrundeliegenden Konzepte und Technologien demonstrieren. Die SuS sollen in der Lage sein, ihre Lösungen logisch darlegen zu können und den entwickelten Code sowie die angewendeten Algorithmen und Strukturen zu erklären. Die Bewertung berücksichtigt sowohl die syntaktische Korrektheit und Lesbarkeit des Codes als auch die Fähigkeit der SuS, ihren Lösungsweg nachvollziehbar darzulegen. Dabei ist es wichtig, dass die SuS selbstständig arbeiten, jedoch wird der sparsame Einsatz von Hilfsmitteln wie KI und externem Code in einem begrenzten Umfang zugelassen, solange das Verständnis und die Kontrolle über die Lösung gewährleistet bleiben.

§ 1 Die SuS sollen den des Codes möglichst selbstständig entwickeln.

§ 2 Der sparsame Einsatz von KI als Debugger ist gestattet, solange die SuS die verwendeten Lösungen verstehen und die zugrundeliegenden Logiken erklären können.

§ 3 Die Verwendung von Fremdcode, der nicht von der SuS stammt, ist teilweise gestattet, jedoch müssen die SuS die Funktionsweise dieses Codes nachvollziehen und erklären können.

§ 4 Der Code-Editor darf die SuS bei grundlegenden Aufgaben durch Vorschläge und Warnungen unterstützen. Der Einsatz eines Debuggers und weiterer Hilfsmittel ist ebenfalls zulässig.

§ 5 Neben der syntaktischen Korrektheit werden die Lesbarkeit und Verständlichkeit des Codes bewertet.

§ 6 Besonders umfangreiche Dokumentationen und eigens entwickelte Tests können ebenfalls zur Bewertung herangezogen werden.

§ 7 Die Wartbarkeit und Modularität des Codes können als ein Bewertungskriterium herangezogen werden.

§ 8 Die SuS sollten in der Lage sein, den verwendeten Code zu erklären, einschließlich der angewendeten Algorithmen und Strukturen.

§ 9 Der Code sollte möglichst in einer Weise geschrieben sein, dass er auch ohne Erklärungen weitgehend verstanden werden kann.

§ 10 Die Bewertung, ob die Kriterien im Einzelfall eingehalten wurden, behält sich die Lehrkraft vor.

3.5 Hohe Selbstständigkeit

Im Fokus dieser Aufgabenstellung steht nicht nur die Entwicklung funktionalen Codes, sondern auch die Fähigkeit, qualitativ hochwertigen, strukturierten und wartbaren Code zu schreiben. Von den SuS wird erwartet, dass sie eine hohe Eigenleistung erbringen und ihren Code selbstständig entwickeln, dabei jedoch auch auf externe Hilfsmittel zurückgreifen können, sofern sie deren Einsatz vollständig nachvollziehen und erklären können.

Neben der Korrektheit und Funktionalität des Codes werden insbesondere die Lesbarkeit, Modularität und Dokumentation bewertet. Die SuS sollen zeigen, dass sie in der Lage sind, die Strukturen und Algorithmen ihres Codes verständlich zu präsentieren und die zugrunde liegende Logik zu erklären. Die Nutzung moderner Tools, wie KI und Erweiterungen eines Code-Editors, wird im begrenzten Rahmen zugelassen, um realistische Entwicklungsbedingungen zu schaffen, dabei bleibt es jedoch die Verantwortung der SuS, alle verwendeten Ansätze vollständig zu verstehen.

§ 1 Die SuS sollen eine hohe Eigenleistung an ihrem Code haben.

§ 2 Der sparsame Einsatz von KI ist gestattet, solange die SuS die verwendeten Lösungen verstehen und die zugrundeliegenden Logiken erklären können.

§ 3 Die Verwendung von Fremdcode, der nicht von der SuS stammt, ist gestattet, jedoch müssen die SuS die Funktionsweise dieses Codes vollständig nachvollziehen und erklären können.

§ 4 Die Wahl des Coding-Editors steht den SuS frei. Jegliche Form von Erweiterungen, wie Code-Vervollständigungen, Debugger oder Syntaxprüfungen sind erlaubt.

§ 1a In Einzelfällen kann die Verwendung bestimmter Werkzeuge und Erweiterungen durch die Lehrkraft untersagt werden.

§ 5 Die Lesbarkeit und Verständlichkeit werden in der Bewertung stark berücksichtigt.

§ 6 Eine grundlegende Dokumentation des Codes durch sinnvolle Kommentare wird erwartet und bewertet. Darauf kann nur verzichtet werden, wenn der Code vollständig und ohne jegliche Hilfe selbsterklärend ist.

§ 7 Die Wartbarkeit, Modularität und Testbarkeit werden bei der Bewertung berücksichtigt.

§ 8 Die SuS müssen in der Lage sein, den verwendeten Code zu erklären, einschließlich der angewendeten Algorithmen und Strukturen. Gegebenenfalls muss die Funktionsweise auch grafisch anhand eines Diagramms oder einer Pseudosprache erläutert werden können.

§ 9 Der Code sollte in einer Weise geschrieben sein, dass er auch ohne Erklärungen weitgehend verstanden werden kann.

§ 10 Die Bewertung, ob die Kriterien im Einzelfall eingehalten wurden, behält sich die Lehrkraft vor.

3.6 Simulation eines realen Softwareentwicklungsprozesses

In diesem Szenario sollen die SuS einen realitätsnahen Softwareentwicklungsprozess simulieren. Ziel ist es, nicht nur funktionale Software zu erstellen, sondern auch professionelle Standards wie Codequalität, Lesbarkeit, Modularität und Testbarkeit zu berücksichtigen. Dabei wird den SuS ein hoher Grad an Freiheit eingeräumt, um moderne Arbeitsweisen und Werkzeuge kennenzulernen und anzuwenden.

Der Einsatz von KI und Fremdcode ist gestattet, allerdings müssen die SuS stets in der Lage sein, die verwendeten Lösungen und deren zugrunde

liegende Logiken verständlich zu erklären. Daraus resultiert, dass der Anteil der Eigenleistung am Gesamtcode zurückgeht. An dessen Stelle treten Aspekte wie die Fähigkeit, komplexe Zusammenhänge in ihrem Code zu dokumentieren und verständlich darzustellen. Die SuS sollen lernen, dass Softwareentwicklung über das bloße Schreiben von Quellcode hinausreicht und diese Erfahrung selbst machen.

§ 1 Die SuS sollen eine hohe Eigenleistung an ihrem Code haben.

§ 2 Der Einsatz von KI ist gestattet, solange die SuS die verwendeten Lösungen verstehen und die zugrundeliegenden Logiken erklären können.

§ 3 Die Verwendung von Fremdcode, der nicht von der SuS stammt, ist gestattet, jedoch müssen die SuS die Funktionsweise dieses Codes vollständig nachvollziehen und erklären können.

§ 4 Die Wahl des Coding-Editors steht den SuS frei. Jegliche Form von Erweiterungen, wie Code-Vervollständigungen, Debugger oder Syntaxprüfungen sind erlaubt.

§ 1a In Einzelfällen kann die Verwendung bestimmter Werkzeuge und Erweiterungen durch die Lehrkraft untersagt werden.

§ 5 Die Lesbarkeit und Verständlichkeit sind zentrale Bewertungskriterien.

§ 6 Eine grundlegende Dokumentation des Codes ist essenziell und ein entscheidendes Bewertungskriterium. Darauf kann nur verzichtet werden, wenn der Code vollständig ohne jegliche Hilfe selbsterklärend ist.

§ 7 Die Wartbarkeit, Modularität und Testbarkeit werden bei der Bewertung im hohen Maß berücksichtigt.

§ 8 Die SuS müssen in der Lage sein, den verwendeten vollständig Code zu erklären, einschließlich der angewendeten Algorithmen und Strukturen. Gegebenenfalls muss die Funktionsweise auch grafisch anhand eines Diagramms oder einer Pseudosprache erläutert werden können.

§ 9 Der Code sollte in einer Weise geschrieben sein, dass er auch ohne Erklärungen vollständig verstanden werden kann.

§ 10 Die Bewertung, ob die Kriterien im Einzelfall eingehalten wurden, behält sich die Lehrkraft vor.

4 Fazit

Die vorgestellten Modelle bieten eine breite Palette möglicher Vorschriften und Bewertungsrastern für Projekte, die individuell angepasst werden können. Sie zielen darauf ab, eine faire und einheitliche Bewertung aller Arbeiten zu gewährleisten und Konflikte, die durch unklare oder fehlinterpretierte Aufgabenstellungen entstehen könnten, zu vermeiden. Durch die klare Struktur dieser Modelle wird sichergestellt, dass die Ergebnisse der SuS unabhängig von ihren unterschiedlichen Vorkenntnissen vergleichbar bleiben. So können sowohl Anfänger als auch Fortgeschrittene unter gleichen Bedingungen ihre Fähigkeiten unter Beweis stellen.

Die beschriebenen Szenarien sind bewusst flexibel gestaltet. Sie dienen einerseits dazu, Orientierung zu bieten, und können andererseits je nach Zielsetzung des Projekts oder Unterrichtsschwerpunkten kombiniert oder angepasst werden.

Ein zentraler Aspekt der vorgestellten Modelle ist die klare Kommunikation der Regeln und Bewertungsmaßstäbe. Es ist essenziell, dass diese des SuS vor Beginn des Projekts vollständig und verständlich vermittelt werden. Auf diese Weise können Missverständnisse vermieden und eine transparente Basis für die Bewertung geschaffen werden. Darüber hinaus fördern klare Vorgaben auch die Selbstständigkeit der SuS, indem sie genau wissen, welche Anforderungen sie erfüllen müssen und wie ihre Leistungen gemessen werden.

Zudem sind die vorgestellten Modelle als Orientierungshilfe zu verstehen und können lediglich einen groben Überblick über mögliche Regelungen geben. Sie sind nicht dazu gedacht, alle denkbaren Szenarien vollständig abzudecken. In der Praxis kann es daher notwendig sein, die Regelungen je nach Anforderungen des Projekts anzupassen. Dies kann bedeuten, dass einzelne Vorschriften verschärft, gelockert oder vollständig überarbeitet werden müssen. Ebenso kann es erforderlich sein, neue Regeln hinzuzufügen, um spezifische Herausforderungen oder Besonderheiten des Projekts zu berücksichtigen.

Ein Aspekt, der in den vorgestellten Modellen nicht vollständig abgedeckt wird, ist beispielsweise die Fragestellung, ob die SuS zu Hause an ihren Projekten weiterarbeiten dürfen oder ob die gesamte Entwicklung ausschließlich im Unterricht stattfinden muss.

In diesem Kontext müssen auch Überlegungen zur Versionsverwaltung des Projekts getroffen werden. Je nach Zielsetzung des Projekts kann dies von einer bloßen Empfehlung, das Projekt zu versionieren, bis hin zu einem vollständigen Verbot der Nutzung einer Versionsverwaltung reichen.

Unter bestimmten Umständen kann es jedoch sinnvoll sein, eine Versionshistorie verpflichtend zu verlangen. Dies könnte insbesondere dazu dienen, den SuS die Möglichkeit zu geben, ihre Eigenleistungen nachzuweisen und die Entwicklung ihres Projekts nachvollziehbar zu dokumentieren. Eine Versionshistorie würde es den Lehrkräften zudem ermöglichen, den Arbeitsprozess und die Fortschritte der SuS transparent zu verfolgen.

Eine solche Regelung könnte auch als Voraussetzung dafür dienen, dass die SuS zu Hause weiter an ihrem Projekt arbeiten dürfen. Die Anforderung, regelmäßig eine Versionshistorie zu pflegen, stellt sicher, dass der Fortschritt kontinuierlich dokumentiert wird und verhindert, dass die Arbeit ausschließlich in unkontrollierbaren Umgebungen stattfindet. Diese Vorschrift bietet sich insbesondere an, da der Code zum Arbeiten von zu Hause ohnehin auf einer Hostingplattform wie GitHub gespeichert werden muss, was die Nutzung einer Versionskontrolle bereits voraussetzt.

Unabhängig von der letztendlichen Entscheidung über die spezifischen Regelungen, sollten diese vor Beginn der Projektarbeit festgelegt und den SuS klar kommuniziert werden.