1 Stundenplaner-Projekt

1.1 Projektbeschreibung

Eine zufällig zusammengesetzte Gruppe von 3 Lernenden (LE) bildet ein Softwareteam. Die Lehrperson (LP) übernimmt die Rolle des Kunden und vermittelt in Form einer mündlichen Sitzung die Bedürfnisse und Erwartungen. Das Softwareteam hat die Aufgabe, die Anforderungen korrekt zu erfassen, in einer geeigneten objektorientierten Sprache umzusetzen (z. B. C#, Java, Python) und nach **12 Lektionen** eine lauffähige und vollständig dokumentierte Lösung zu präsentieren.

Hinweis (siehe auch Zusatzhinweise)
Während der Entwicklung kann die LP (Kunde) Änderungswünsche einbringen. Das Team muss solche Änderungen protokollieren und begründen, wie sie umgesetzt oder abgelehnt wurden.

1.2 Fachliche Anforderungen

1.2.1 Ziel

Ein **automatisch generierter Stundenplan** für ein Semester soll erzeugt werden. Dabei werden **Ressourcen** (Schüler:innen, Lehrpersonen, Räume, Fächer) **unter Berücksichtigung von Verfügbarkeiten** kombiniert.

1.2.2 Grundfunktionen

- Verwaltung von **Schüler:innen, Lehrpersonen, Räumen, Fächern**
- Definition von **Verfügbarkeiten** (z. B. Lehrperson: Dienstagnachmittag)
- **Zuweisung**: Schüler besuchen Fächer, Lehrpersonen können bestimmte Fächer unterrichten
- **Automatische Stundenplanerstellung** nach definierten Kriterien
- **Persistente Datenspeicherung** (nach Neustart wiederherstellbar)

1.2.3 Qualitätskriterien bei der Planung

- Fächer an **Randzeiten** werden **schlechter bewertet**
- **Zwischenstunden** werden **schlecht bewertet**
- Ein **ressourcenschonender Stundenplan** (weniger Räume parallel genutzt) wird **besser bewertet**
- Die **Gewichtung** dieser Kriterien soll **in der Applikation einstellbar** sein

1.3 Dokumentation (Pflichtbestandteile der Abgabe)

Die Dokumentation ist Teil der Bewertung und muss vollständig im Git-Repository abgelegt werden. Sie dient zur Nachvollziehbarkeit der Teamarbeit, der technischen Umsetzung und der Reflexion.

Kapitel	Inhalt
1. Deckblatt	Titel, Teammitglieder (Name, Klasse), Abgabedatum, Modul
2. Projektbeschreibung	Kurze Zusammenfassung des Projekts und der Ziele
3. Anforderungen & Use-Cases	Kundenbedürfnisse, Use-Case-Beschreibung, ggf. Szenarien
4. Analyse & Planung	UML-Diagramme (siehe unten), Architekturentscheidungen, Datenmodell

5. Umsetzung	Beschreibung der wichtigsten Klassen, Methoden und Abläufe
6. Bedienungsanleitung	Wie wird das Programm ausgeführt und getestet
7. Qualitätskriterien & Bewertung	Wie werden Randzeiten, Zwischenstunden usw. bewertet
8. Tests & Beispielausgabe	Beispiel für generierten Stundenplan, Screenshots
9. Reflexion & Teamarbeit	Was lief gut, was würde das Team verbessern
10. Arbeitsrapport / Zeitaufwand	Siehe Vorlage unten

1.3.1 UML-Diagramme (Pflichtbestandteil vor Coding-Beginn)

Das Team erstellt mindestens **zwei UML-Diagramme** in der Planungsphase:

1. **Klassendiagramm** (zentrale Klassen, Beziehungen, Attribute, Methoden)
2. **Programmablaufplan** oder **Use-Case-Diagramm** (Benutzerinteraktionen mit der Anwendung)
3. **Optional empfohlen: Sequenzdiagramm** für den Ablauf der Plan-Generierung

Diese Diagramme müssen **vor Beginn der Programmierung** im Git-Repository unter der Struktur docs/uml/ abgelegt sein.

1.4 Ablauf & Organisation

Phase	Inhalt	Zeitrahmen
1. Kick-Off & Kundenbriefing	**LP erklärt Bedürfnisse (mündlich)**	**0.5 Lektion**
2. Anforderungsanalyse & UML-Planung	**Git-Repository, Diagramme, Datenmodell, Aufgabenverteilung**	**2.5 Lektionen**
3. Implementierung (Coding)	**Entwicklung der Applikation**	**6–7 Lektionen**
4. Test & Optimierung	**Bewertung, Kriterien Tests, Dokumentation**	**2–3 Lektionen**
5. Abschlusspräsentation	**Demo + Review + Abgabe**	**ab 12 Lektion**

1.5 Arbeitsrapport & Aufgabenverteilung

Jede Gruppe führt einen Arbeitsrapport. Dieser dient zur Leistungsbewertung und Transparenz, wer welche Teile bearbeitet hat.

Beispielhafte Vorlage (kann als Tabelle in Markdown, Excel oder PDF geführt werden)

Datum	Team-mitlied	Aufgabe / Arbeitspaket	Zeit (h)	Status	Bemerkung
01.11.2025	Lara M.	UML-Klassendiagramm erstellt	1.5	✔	Abgelegt in `/docs/UML`
01.11.2025	Tim M.	Teacher, Student Klassen implementiert	2.0	✔	Git-Commit #15
02.11.2025	Kevin R.	JSON-Speicherfunktion programmiert	1.0	✔	Testdaten gespeichert
03.11.2025	Lara M.	Bewertungslogik getestet	1.5	⏳	Anpassung nötig
04.11.2025	Gruppe	Meeting zur Aufgabenverteilung	0.5	✔	Protokoll vorhanden

Vorgabe: Der Rapport sollte **mind. 1× pro Woche aktualisiert** werden. Er dient auch zur **Beurteilung der Teamleistung** und **Einzelleistung**.

1.6 Abgabe

Abgabe über Git-Repository (Zugriff muss der Lehrperson zu Beginn des Projekts gewährt werden), das Folgendes enthalten muss:

📁 Projektstruktur (Beispiel)

```
TimetablePlanner/
├─ src/                    # Quellcode
├─ docs/
│   ├─ UML/                # Klassendiagramm, Use-Case
│   ├─ Projektbeschreibung.pdf
│   ├─ Benutzerhandbuch.pdf
│   ├─ Arbeitsrapport.xlsx / .md
│   └─ Reflexion.md
├─ README.md               # Kurzbeschreibung, Startanleitung
└─ .gitignore
```

1.7 Zusatzhinweise

- Änderungen des «Kunden» (LP) müssen im **Arbeitsrapport** dokumentiert werden («Change Request»).
- Die Applikation kann eine **Konsolenanwendung** sein.
  Eine GUI ist **nicht erforderlich**, aber erlaubt.
- Für die Persistenz dürfen **Text-, JSON-, XML- oder SQLite-Dateien** verwendet werden.
- Testdaten müssen enthalten sein, um die Stundenplan-Generierung zu demonstrieren.