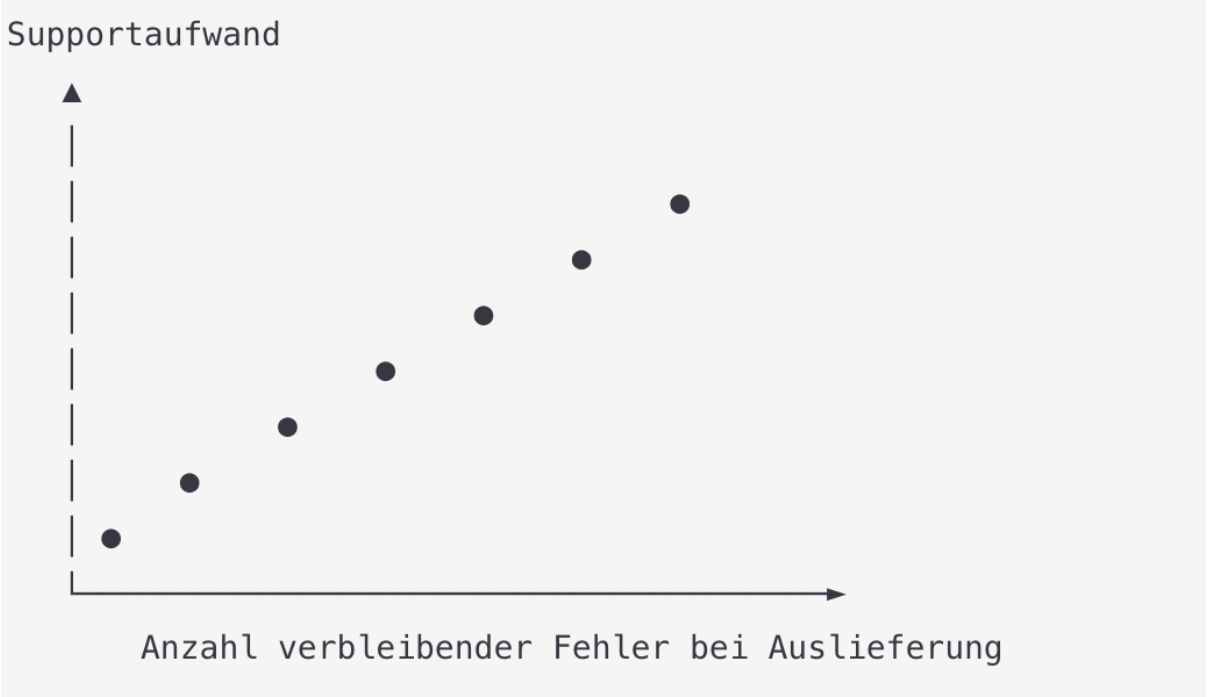
Blatt 9 und 10

Defects in software are related with the **cost, effort, and intensity of product support**

## Why defects at delivery increase product support load:

- Remaining defects are bugs still in the software when it's released.
- More defects = more support work: users report issues, developers must fix them.
- This costs time, money, and frustrates users.
- After a certain point, the support effort grows very quickly with each added defect.
- Fewer defects mean: less stress, happier users, lower costs.




Unternehmen wollen die Supportkosten gering halten und zufriedene Kunden haben. Wenn viele Fehler im Produkt sind:

- Müssen Support-Teams Überstunden machen.
- Entwickler werden von neuen Projekten abgezogen.
- Kunden sind unzufrieden und wechseln zur Konkurrenz.

Darum ist Qualitätssicherung vor dem Release entscheidend!

Ex2

No

**Bias and blind spots**
Developers might unconsciously avoid testing scenarios where they expect problems.
**Lack of independence**
Testing by someone else (e.g. QA) is more objective and often uncovers more issues.**T unnel vision**
Developers might focus too much on "happy paths" and overlook real-world usage patterns.
**Time pressure**
Under deadlines, testing may be rushed or skipped altogether if it's not an external role.
**Limited user perspective**
Developers often don’t test like real users would, so usability and real-life workflows may go unchecked.

Yes

**Deep understanding of the code**
Developers know the logic and structure best, so they can test edge cases effectively.
**Faster feedback loop**
Issues can be found and fixed immediately during development without delays.
**Improved code quality**
Writing tests makes developers think more carefully and write cleaner, more modular code.
**Cost-efficient**
Early testing by developers reduces the cost of later bug fixes and avoids expensive rework.
**Responsibility mindset**
Encourages accountability—developers are more likely to ensure their code works properly.

Ex4

“Did the old features still work after I changed something?”
**Regression testing** is a type of software testing that checks whether **new changes or updates** in the code have accidentally broken **existing functionality**.

**Automated tests** (unit, integration, UI) are commonly used for regression testing.
They are re-run **after every change**, like in CI/CD pipelines.

Ex3

## 1. URL Validation

**Valid URL:** z. B. `https://example.com`
**Invalid URL:** z. B. `http:/invalid-url`

## 2. Notification Preferences

**a) Frequency: Valid frequencies:** daily, weekly, monthly
**b) Communication Channe: Valid channels:** email, sms

## 3. Subscription Management

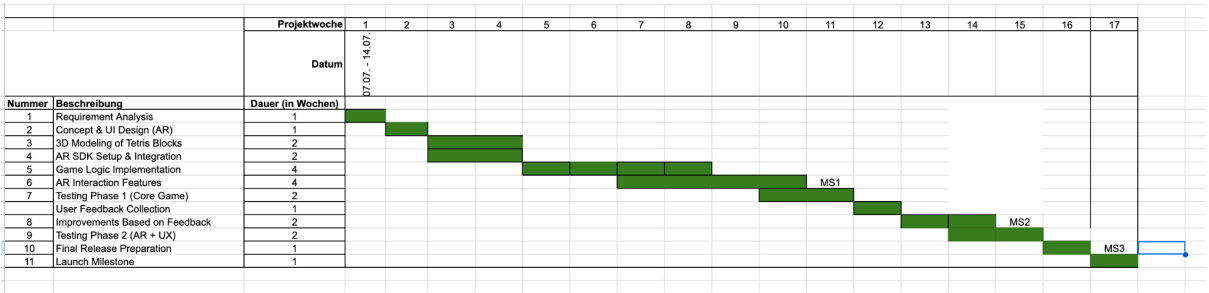
**Valid modifications:** Änderung des Plans, z. B. Upgrade oder Downgrade
Gültige Kündigung

Ex5

| Feature | Black Box Testing | White Box Testing |
|---|---|---|
| Focus | Tests **functionality** | Tests **internal logic, code, paths** |
| Knowledge required | No knowledge of code needed | Requires **programming/code knowledge** |
| Tester role | Usually **QA/tester** | Usually **developer/unit tester** |
| Examples | UI tests, API tests, system tests | Unit tests, branch coverage, loop testing |
| What is tested? | Inputs → Outputs | Control flow, conditions, loops, paths |

Übung10

Ex1



Ex2

# 3. How would you staff the project to achieve minimal time to market?

🎯 Goal: **Fast delivery** through **parallel work**, **specialization**, and **early testing**.

| Role | Responsibilities |
|---|---|
| **Project Manager** | Planning, coordination, communication |
| **2 AR Developers** | AR SDK integration, Unity setup, spatial mapping |
| **1 Game Developer** | Core game logic (Tetris mechanics, game states) |
| **1 3D Designer** | Creating 3D Tetris blocks and visual assets |
| **1 UI/UX Designer** | Designing AR UI, gestures, voice/blick interaction |
| **1 QA Engineer** | Testing game mechanics and AR interactions |
| **1 DevOps/Build Engineer** | Build pipelines, deployment, testing environments |

Ex3

## Recommended: Agile / Scrum

## ✅ Why Agile?

- **Short sprints (1–2 weeks)** → fast feedback and progress

- **Early user testing** of AR interactions and controls
- **High flexibility** for changing priorities or adding AR features
- Fits well for **innovative and experimental projects** like AR games

✅ Alternative: **Kanban** for continuous flow
❌ Not recommended: **Waterfall**, as it's too rigid and slow for iterative AR development

Ex4

# Your project is already over time and over budget. What can you do?

## 🛠️ 5 Options to Finish the Project:

1. **Reduce Scope (Feature Cut)**
   → Focus on the **core gameplay**; postpone extras (e.g., multiplayer, skins)

2. **Add Resources**
   → Bring in **additional developers or freelancers**, especially for AR or 3D design

3. **Postpone Technical Debt**
   → Prioritize a **working MVP**, clean up code and refactor later

4. **Outsource Specific Work**
   → Delegate **UI prototypes or 3D assets** to external teams

5. **Phased Release Strategy**
   → Launch a **minimal version first**, and roll out features in updates