

# System

Modul A

Modul



je -10

!

!



je -6



?



je -4

?



je -2

?

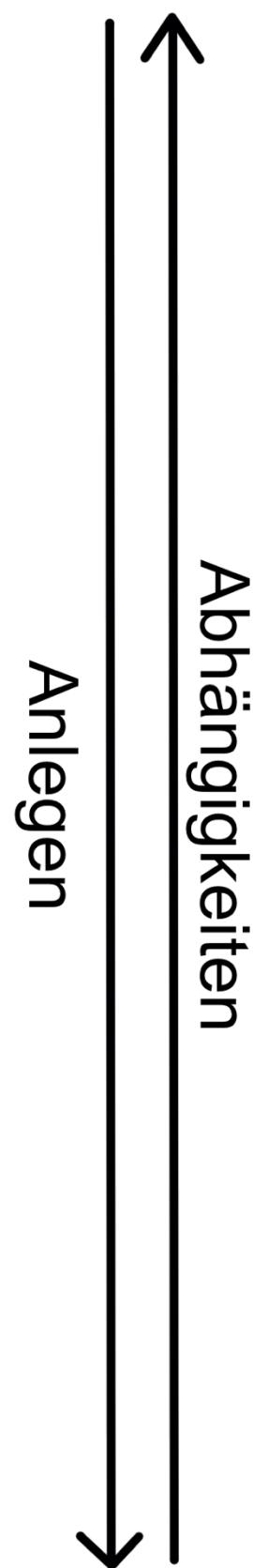
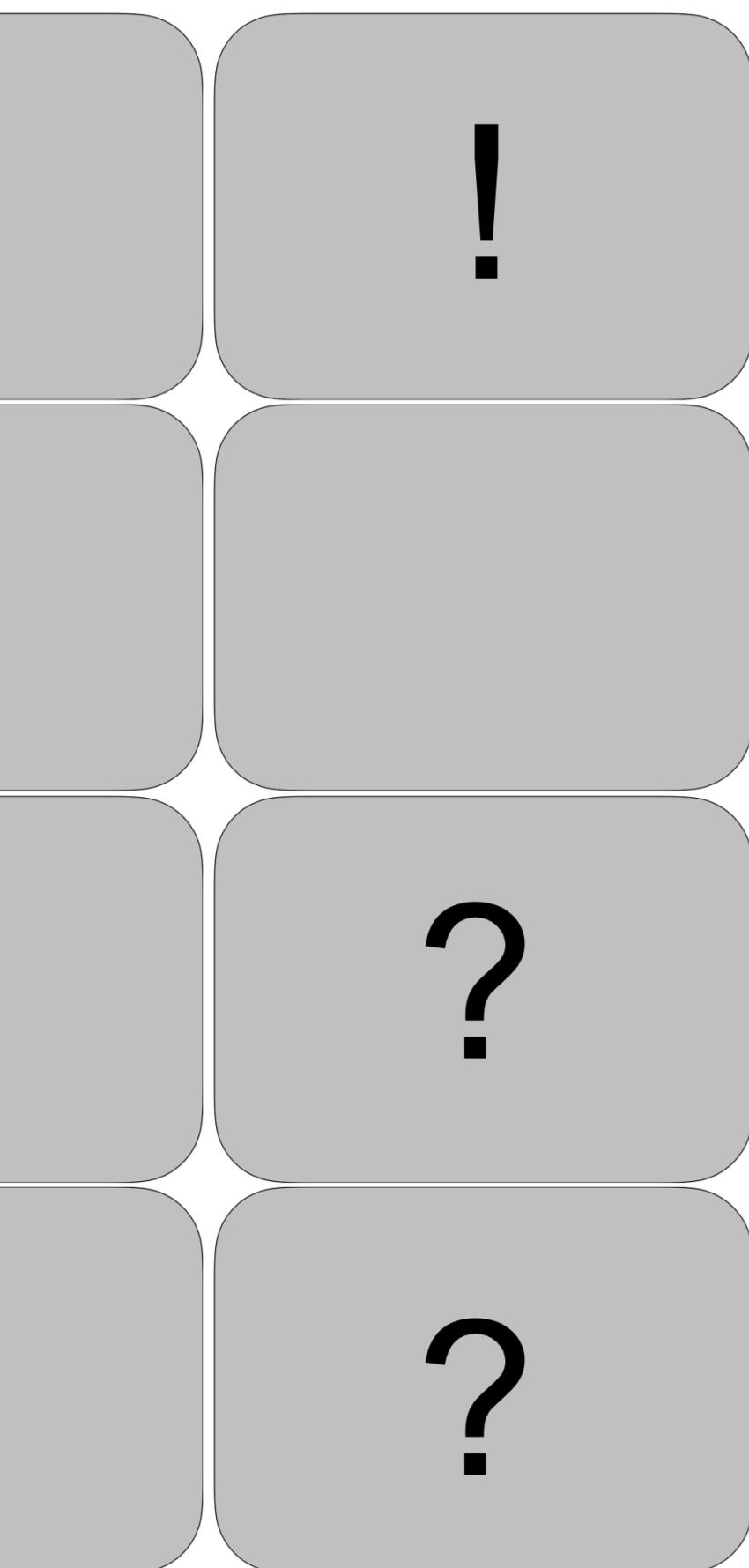


+10 bei Abschluss

+10 bei Absc

B

## Modul C



Anlegen

Abschluss

+10 bei Abschluss

# System

Modul A

Modul



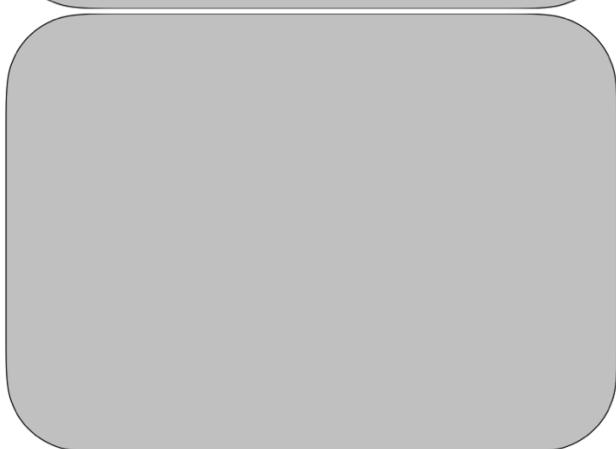
je -10

!

!



je -6

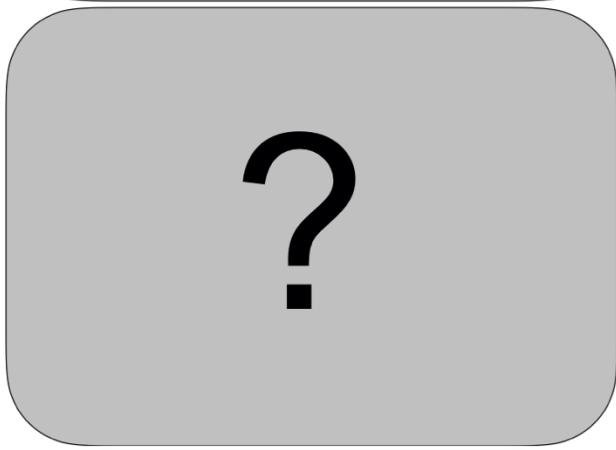


?



je -4

?

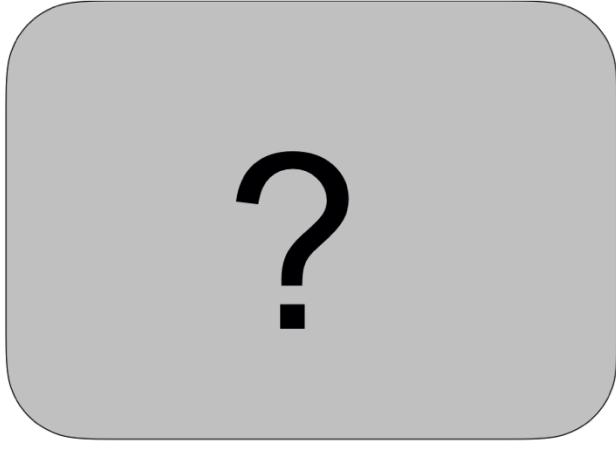


?



je -2

?



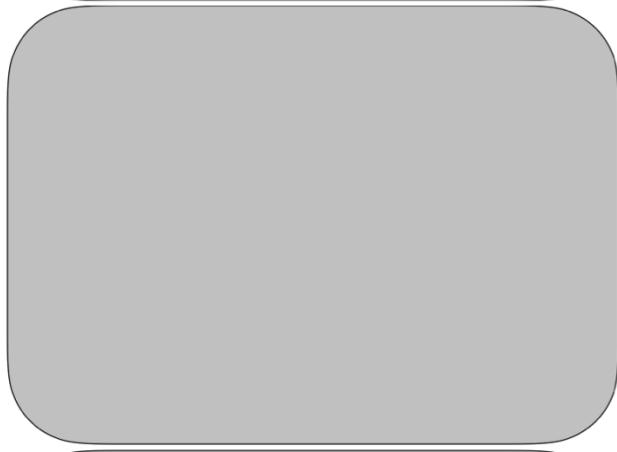
?

+10 bei Abschluss

+10 bei Absc

B

## Modul C



Anlegen

Abhängigkeiten



Abschluss

+10 bei Abschluss

# Nutzeranzeige

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179

## Endwertung

	Nutzeranzahl 	abgeschlosse ne Module	TS in Architektur (Reihe 1) 	TS in Feature (Reihe 2) 	TS in Feature (Reihe 3) 	TS in Feature (Reihe 4) 	Plus- Punkte	Minus- Punkte	Gesamt
Team 1		x10=	x10=	x6=	x4=	x2=			
Team 2		x10=	x10=	x6=	x4=	x2=			

# Übersicht

## a) Ticket bearbeiten



\* mit zwei Würfeln würfeln:

- TS aufnehmen (opt.) und →
- Pasch? →
- Pro getroffenen Würfel →

### - Ticket fertig?:

- Aktuelles Ticket im System anlegen
- !/?-Feld?: verdeckt !/?-Karte ziehen
- Punkte vermerken. (Nur bei )
- Neues Ticket. Arbeitsstelle platzieren
- !/?-Karte vorlesen und ?-Karte direkt ausspielen.

!-Karte darf in einem vorteilhaften Moment am Ende eines Zuges ausgespielt werden.

Aktuelles Ticket



## ODER

## b) TS abbauen

- Arbeitsstelle auf TS abbauen umdrehen
- TS bestimmen
- Mit einem Würfel würfeln
  - In : 4, 5 oder 6
  - In : 5 oder 6
- geschafft?: TS abbauen → und auf Arbeitsstelle umdrehen.

# Übersicht

## a) Ticket bearbeiten



\* mit zwei Würfeln würfeln:

- TS aufnehmen (opt.) und →
- Pasch? →
- Pro getroffenen Würfel →

Aktuelles Ticket



### - Ticket fertig?:

- Aktuelles Ticket im System anlegen
- !/?-Feld?: verdeckt !/?-Karte ziehen
- Punkte vermerken. (Nur bei )
- Neues Ticket. Arbeitsstelle platzieren
- !/?-Karte vorlesen und ?-Karte direkt ausspielen.

!-Karte darf in einem vorteilhaften Moment am Ende eines Zuges ausgespielt werden.

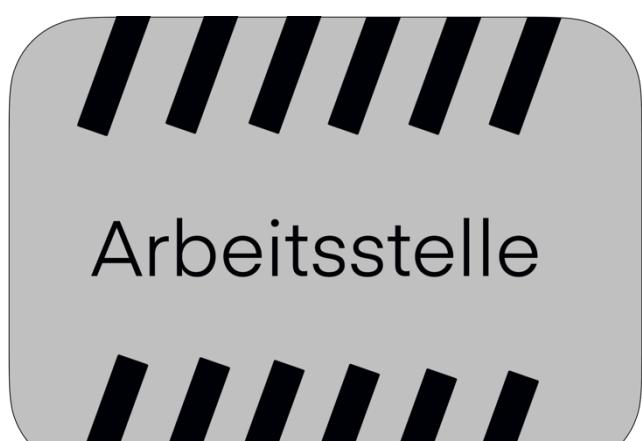
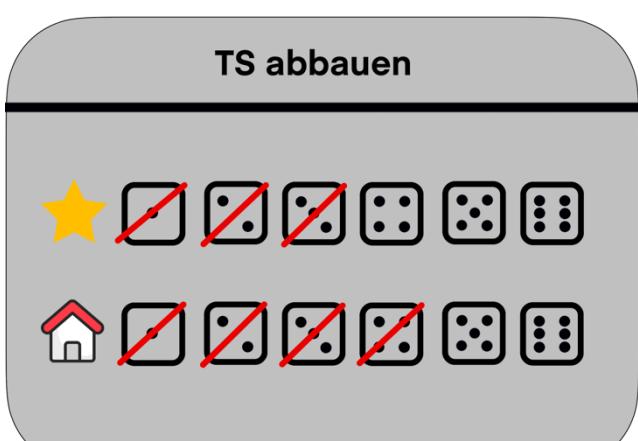
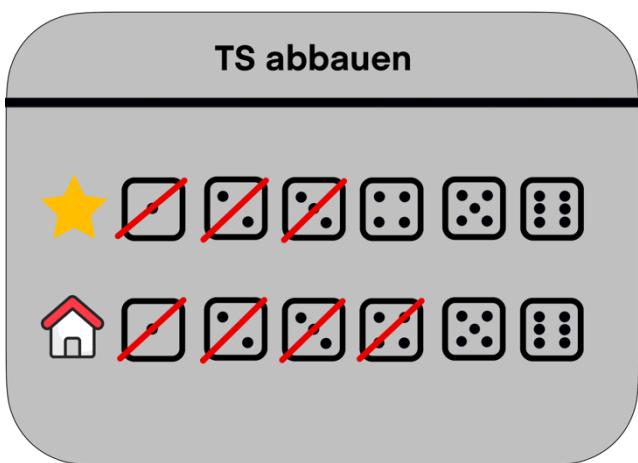
## ODER

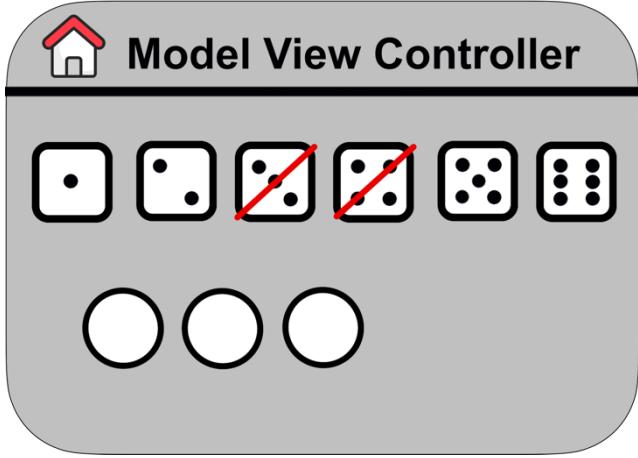
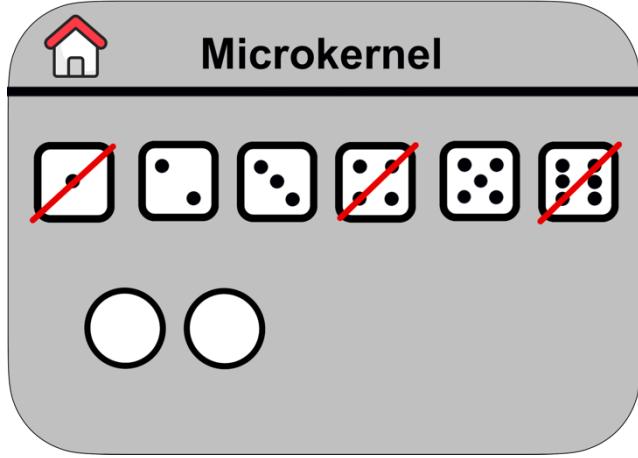
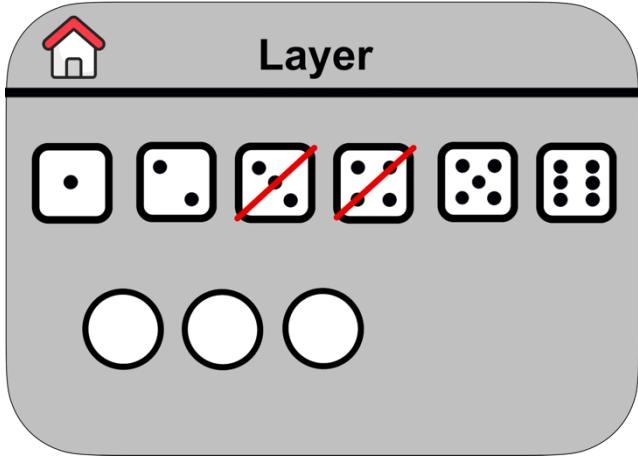
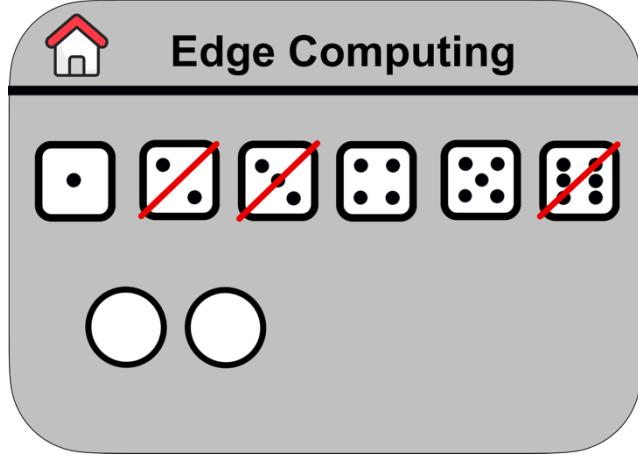
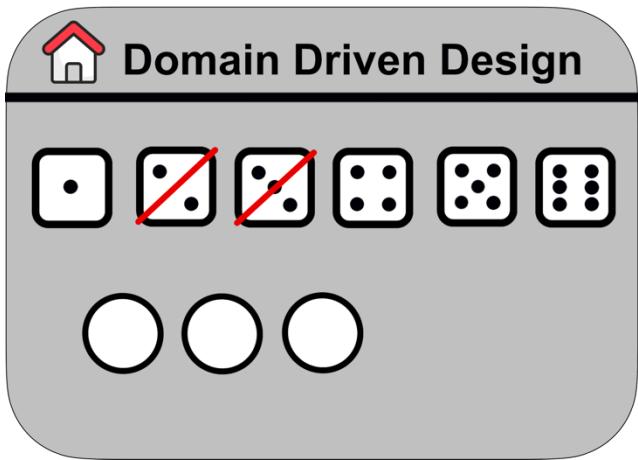
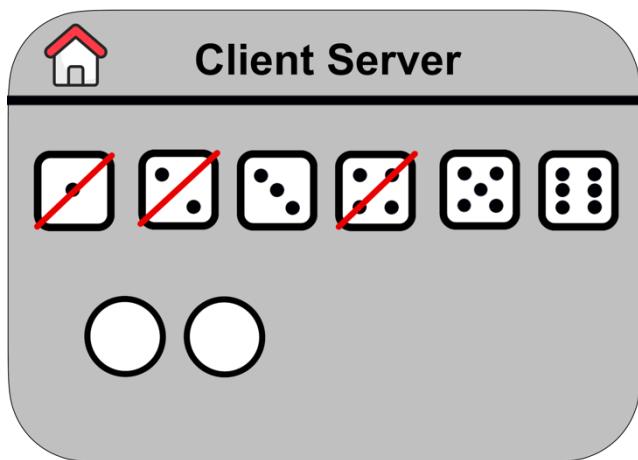
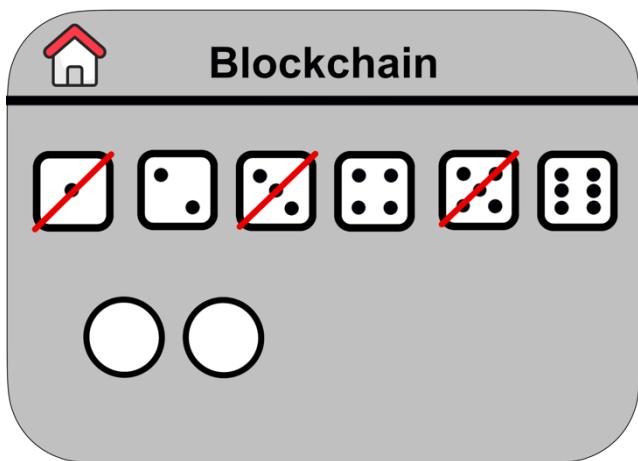
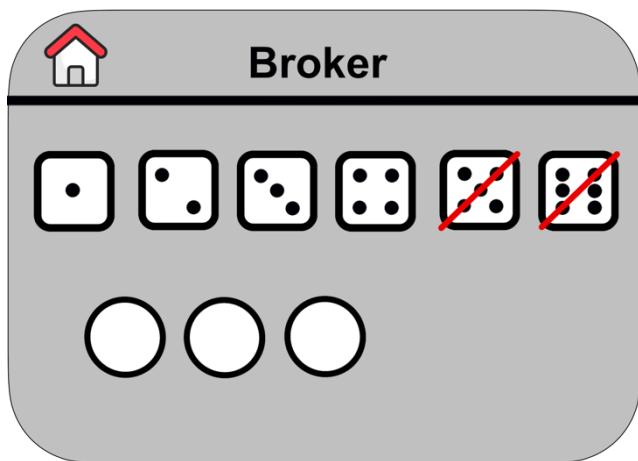
## b) TS abbauen

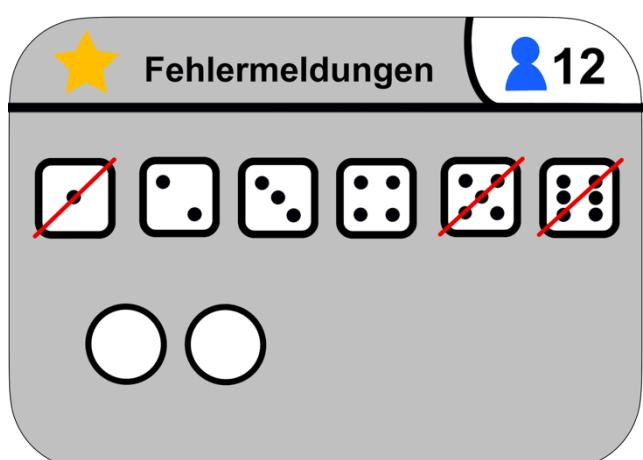
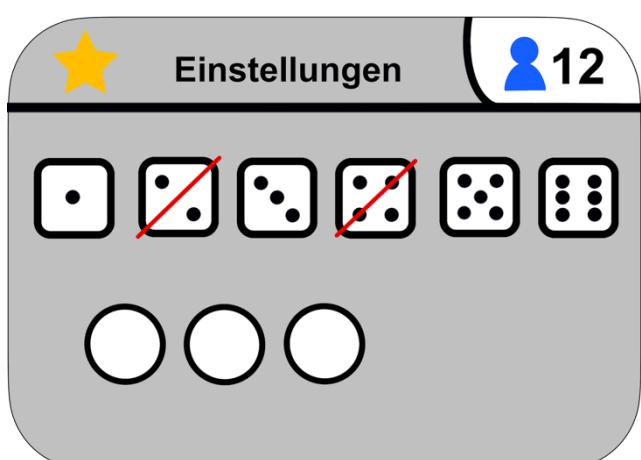
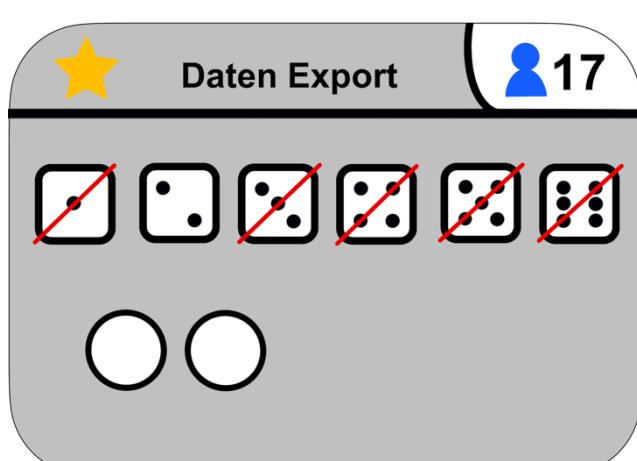
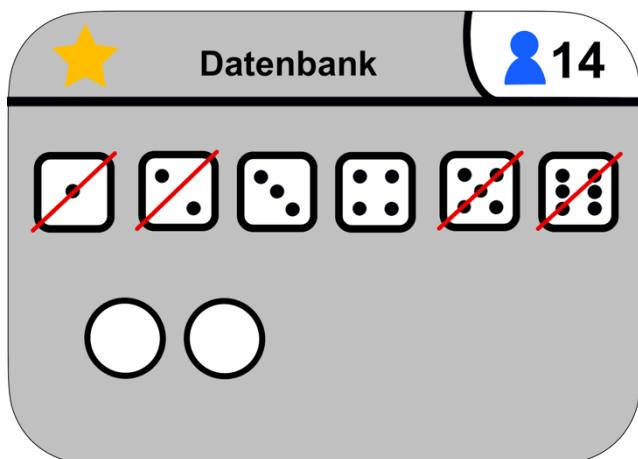
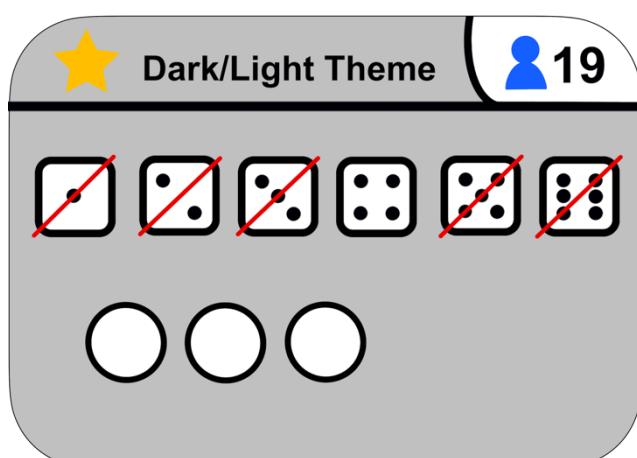
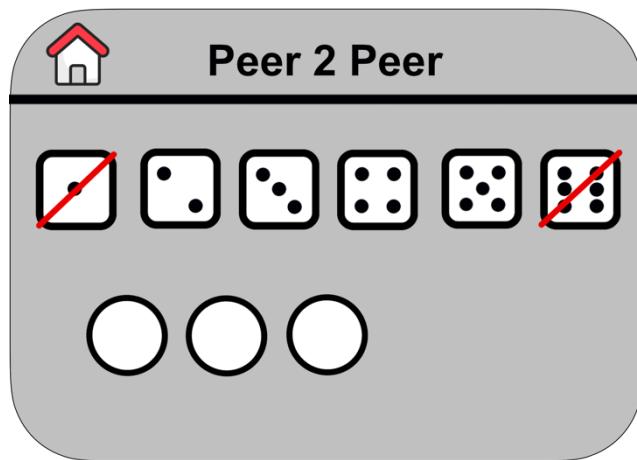
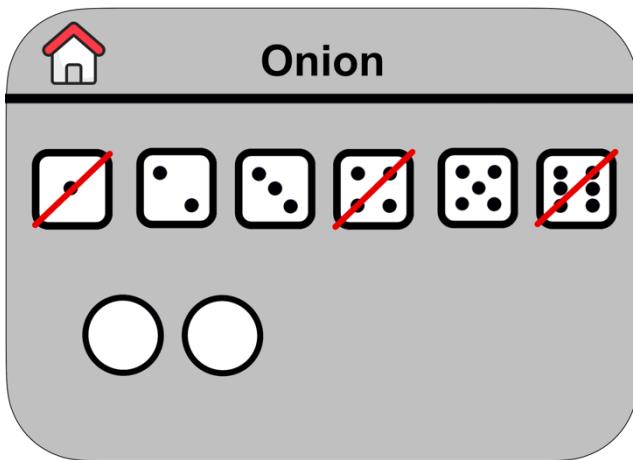
- Arbeitsstelle auf TS abbauen umdrehen
- TS bestimmen
- Mit einem Würfel würfeln
  - In : 4, 5 oder 6
  - In : 5 oder 6
- geschafft?: TS abbauen → und auf Arbeitsstelle umdrehen.

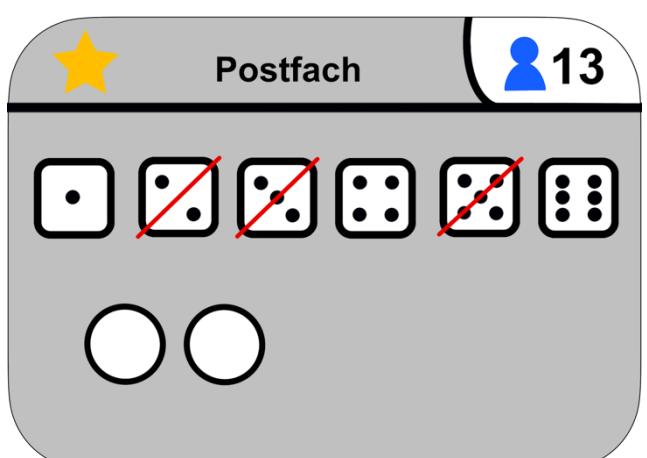
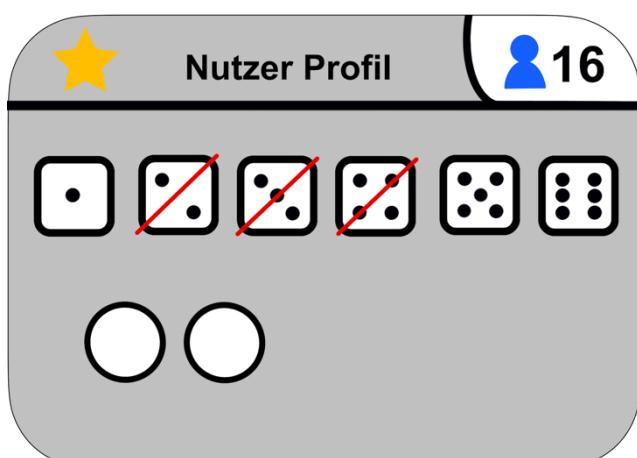
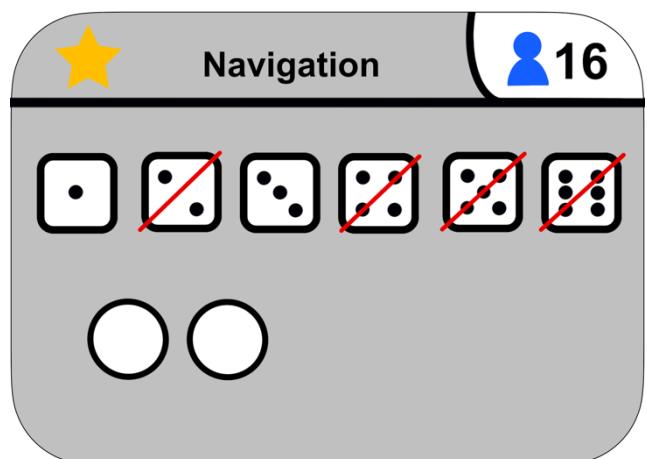
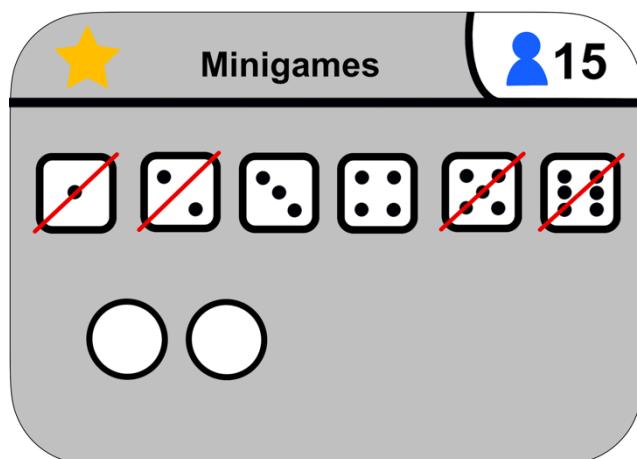
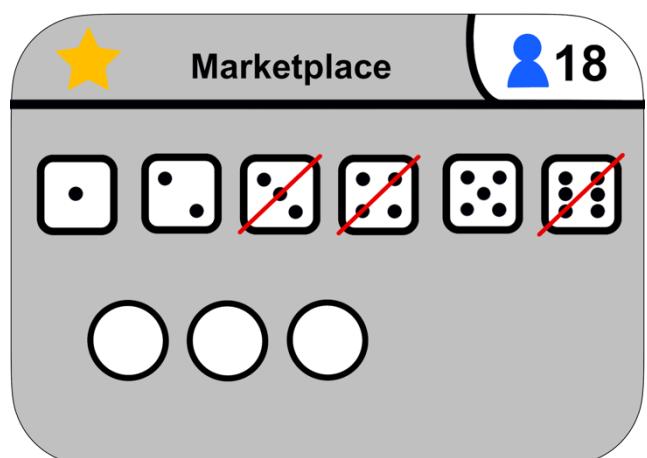
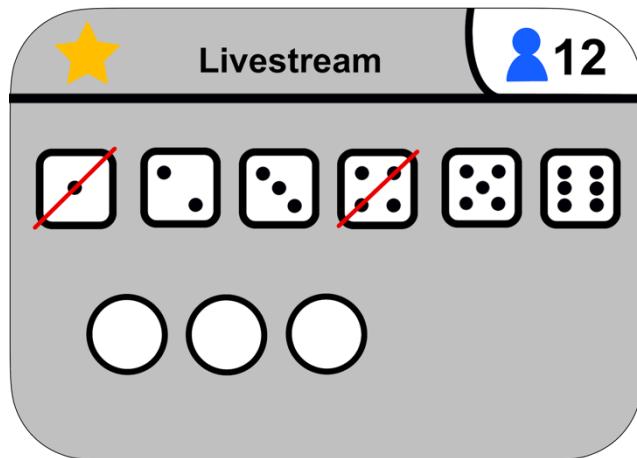
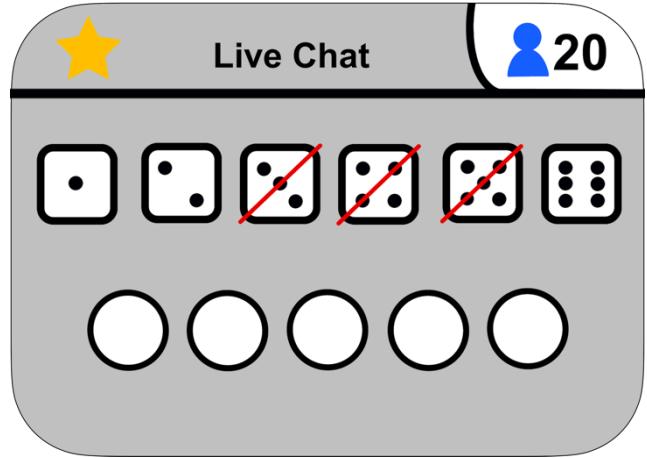
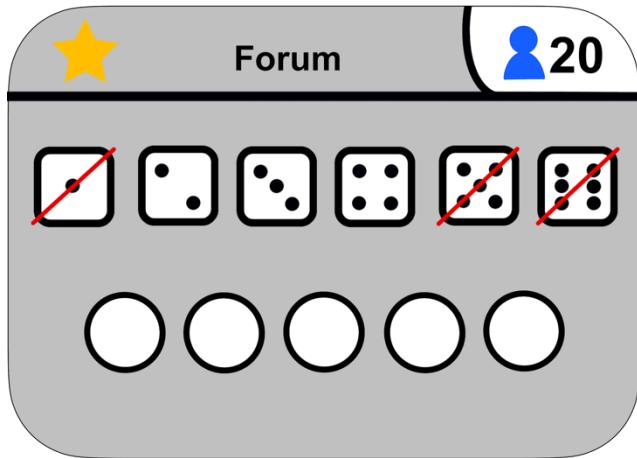
## Faltlinie

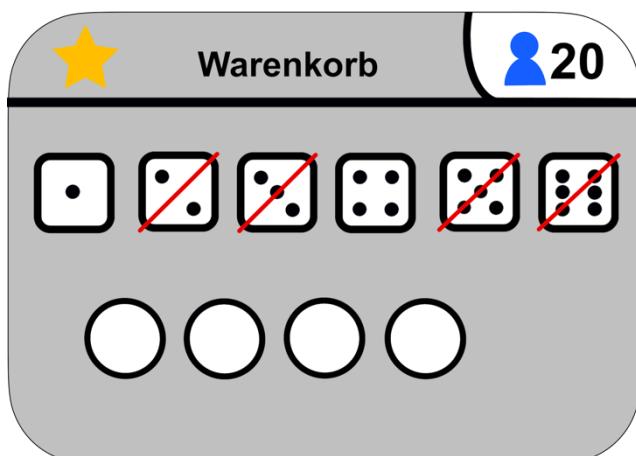
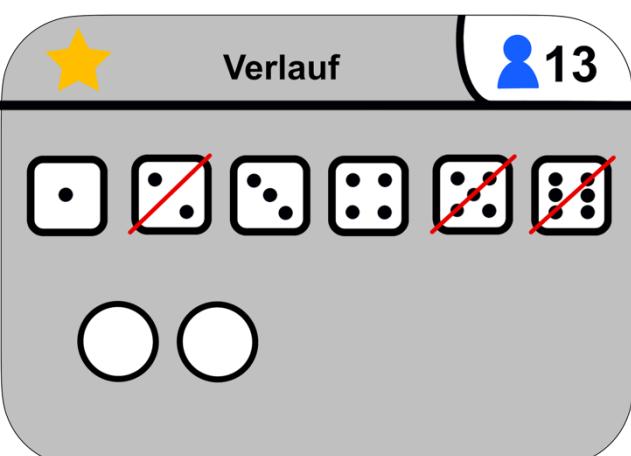
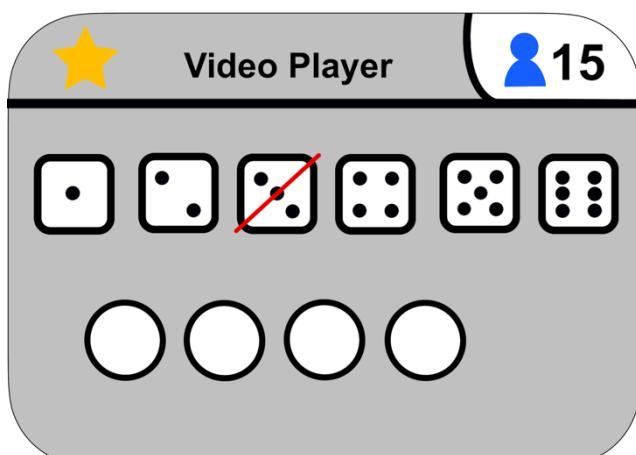
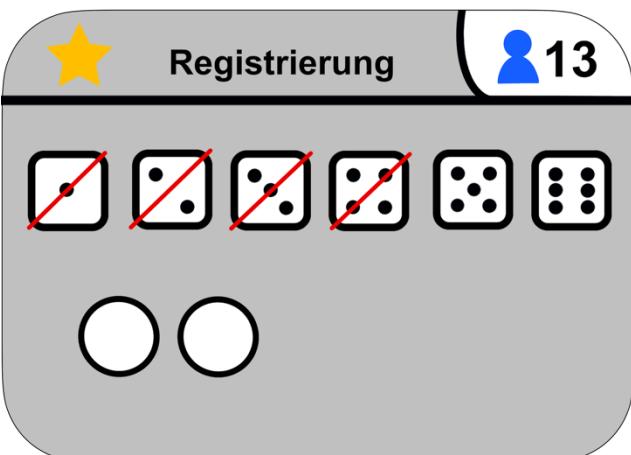
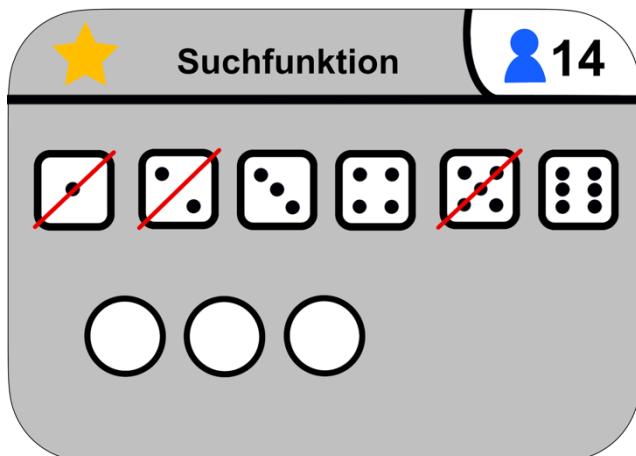
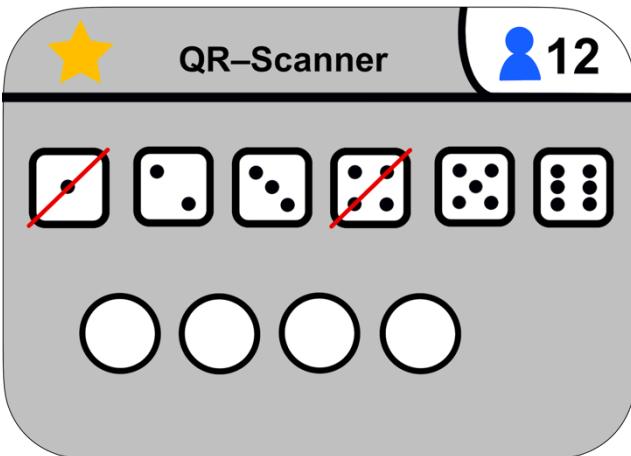
---











!

### Team-Kommunikation

Team-Kommunikation kann Mitgliedern bewusst machen, was für Auswirkungen Technische Schulden haben und dass man sie effektiv managen sollte.

Du darfst eine beliebige TS entfernen.

!

### Refactoring

Ein Entwickler wird auf das angemessene Aufteilen von zu langen Klassen und Methoden gesetzt. Dies verringert die Komplexität des Projekts.

Du darfst eine beliebige TS entfernen.



## Ursprung aufdecken

Das wiederholte Fragen nach dem Grund hilft dabei, den Ursprung von Technischen Schulden aufzudecken.

**Du darfst eine beliebige TS entfernen.**



## Werkstudent einstellen

Das Management beschließt einen Werkstudenten einzustellen. Dieser kann einfache Aufgaben übernehmen, sodass die Entwickler mehr Zeit für anspruchsvolle Aufgaben haben.

**Du darfst einen weiteren Spielzug machen.**



## Die W-Fragen klären

Das richtige Erkennen von Technischen Schulden mittels der W-Fragen, hilft dabei von Symptomen auf die eigentlichen Technischen Schulden zu schließen. Es fällt leichter, Technische Schulden zu beseitigen.

**Du darfst eine beliebige TS entfernen.**



## Mehr Personal

Das Management erkennt, dass viele Entwickler ausgelastet sind. Um die Arbeitslast besser aufzuteilen, beschließt das Management neue Entwickler einzustellen.

**Du darfst einen weiteren Spielzug machen.**



## Gute Einbindung

Durch Einbindung in die Probleme von Programmierern ist das Management über aktuelle Probleme in der Weiterentwicklung aufgeklärt. Das Management vertraut den Entwicklern und ist bereit mehr Zeit in Qualität zu investieren.

**Du darfst einen weiteren Spielzug machen.**



## Spezialisten einschalten

Durch das Hinzuholen von Spezialisten können vorher unbekannte Probleme aufgedeckt und behoben werden.

**Du darfst eine beliebige TS entfernen.**



## Erfahrungswissen

Durch Kommunikation mit Programmierern aus anderen Projekten kann der Ursprung von Technischen Schulden aus Erfahrungswissen ermittelt und behoben werden.

**Du darfst eine beliebige TS entfernen.**



## Dieses Spiel spielen

Durch das generelle Aufklären von Teammitgliedern über Technische Schulden kann eine unvorteilhafte Priorisierung und das Scheitern des Projekts verhindert werden.

**Du darfst einen weiteren Spielzug machen.**

## ? Fehlendes Bewusstsein

Das Management missversteht technische Schulden als vorübergehenden Trend und verpasst die Gelegenheit, sich mit der Thematik auseinanderzusetzen.

**Dein Gegner darf eine TS auf dein aktuelles Ticket legen.**

## ? Urlaub

Es fehlt die Zeit, den Code für das aktuelle Ticket zu dokumentieren. Der Entwickler, der sich als einziger mit dem aktuellen Ticket auskennt, ist im Urlaub. Du musst daher auf die Rückkehr des Entwicklers warten.

**Falls du zwei oder mehr TS im System hast, setze eine Runde aus.**

## ? Fehlendes Testframework

Das Einrichten des Testframeworks kostet zu viel Zeit, daher wird vorerst darauf verzichtet. Jede neue Zeile Code erzeugt nun technische Schulden, weil die Tests später nachträglich erstellt werden müssen, sobald ein Framework vorhanden ist.

**Dein Gegner darf eine TS auf ein beliebiges Feature-Ticket legen.**

## ? Erhöhte Kosten

Technische Schulden führen zu Performance-Einbußen. Um die Performance zu optimieren, werden die Server parallelisiert, was zu zusätzlichen Lizenzkosten führt.

**Falls du zwei oder mehr TS in deinem System hast, darf dein Gegner eine TS auf ein beliebiges Feature-Ticket legen.**

## ? Äußerer Teufelskreis

Es gibt zu wenig Entwickler im Unternehmen, was dazu führt, dass sie zu viele Aufgaben erledigen müssen. Einige Entwickler sind dadurch gestresst und verlassen das Unternehmen.

**Jedes Ticket in deinem System, was zwei oder mehr TS hat, bekommt eine weitere TS, die dein Gegner platzieren darf.**

## ? Unüberschaubare Test-Schulden

Die angesammelten Test-Schulden sind nicht mehr überschaubar. Instabile oder veraltete Tests mindern das Vertrauen in die Testergebnisse und führen dazu, dass Fehlermeldungen ignoriert werden. Dadurch werden wichtige Fehler übersehen, was zum Absturz des Systems führt. Das Team muss intensiv Fehler suchen und hat keine Zeit, neue Features zu implementieren.

**Du setzt eine Runde aus.**

## ? Kontrollverlust

Durch das Überdenken von zu viel Eventualitäten in der Planphase wurden zu viele Optionen implementiert, die noch nicht benötigt werden. Das Projekt ist nur noch schwer überschaubar. Das Management bemüht sich, die Kontrolle wieder zu bekommen.

**Falls du zwei oder mehr TS in deinem System hast, darf dein Gegner eine TS auf ein beliebiges Feature-Ticket legen.**

## ? Verkettete Ursachen

Wegen Zeitdruck mangelt es an Dokumentation. Die fehlende Dokumentation wiederum macht es neuen Entwicklern schwer sich in dem System zurecht zu finden. Sie verursachen dadurch weitere Technische Schulden.

**Jedes Ticket in deinem System, das zwei oder mehr TS hat, bekommt eine weitere TS, die dein Gegner platzieren darf.**

## ? Stress und Kündigungen

Das Management hat technische Schulden verursacht. Diese sollen nicht zurückgezahlt werden. Dadurch haben Entwickler langfristig mehr zu tun und sind gestresst. Einige verlassen deswegen sogar das Unternehmen.

**Falls du zwei oder mehr TS im System hast, setze eine Runde aus.**

## ? Schlechte Anforderungen

Schlecht dokumentierte Anforderungen vom Kunden führen dazu, dass die Architektur falsch gewählt wurde.

**Dein Gegner darf eine TS auf ein beliebiges Architektur-Ticket legen.**

## ? Kommunikationsprobleme

Der Kunde fordert eine Umpriorisierung des aktuellen Tickets. Das Management teilt den Entwicklern dies allerdings zu kurzfristig mit. Sie müssen für die rechtzeitige Implementation Technische Schulden aufnehmen.

**Tausche dein aktuelles Ticket gegen ein neues Ticket aus der Mitte und belege dieses sofort mit einer TS**

## ? Unsichtbare Schulden

Das System funktioniert nach außen hin fehlerfrei, obwohl bereits viele Technische Schulden aufgenommen wurden. Da es aktuell keine Beschwerden vom Kunden gibt, entscheidet das Management, weiterhin an Abkürzungen in der Implementation festzuhalten.

**Dein Gegner darf eine TS auf dein aktuelles Ticket legen.**

## ? Veraltete Technologie

Die Technologie, für welche sich das Management zu Beginn entschieden hat, ist mittlerweile in die Jahre gekommen. Trotzdem möchte das Management keine Aktualisierung durchführen. Das Implementieren von neuen Features dauert immer länger.

**Dein Gegner darf eine TS auf ein beliebiges Feature-Ticket legen.**

## ? Finanzieller Druck

Das aktuelle Ticket benötigt zusätzliche Lizenzkosten und sprengt damit den finanziellen Rahmen des Projekts. Um das Budget einzuhalten, muss eine billige und provisorische Lösung gefunden werden.

**Dein Gegner darf eine TS auf dein aktuelles Ticket legen.**

## ? Fehlende Funktionen

Der Kunde beklagt sich über fehlende Funktionen im System. Die Funktionalität kann aufgrund von zu vielen Technischen Schulden nicht implementiert werden.

**Falls du drei oder mehr TS im System hast, darf dein Gegner eine TS auf ein beliebiges Architektur-Ticket legen.**

## ? Das will ich nicht mehr!

Der Kunde hat bei einer Demo gesehen, wie das neueste Ticket aussehen wird und will es nicht mehr. Stattdessen soll ein anderes Ticket implementiert werden und das möglichst schnell. Es müssen gezwungenermaßen Technische Schulden aufgenommen werden.

**Tausche dein aktuelles Ticket gegen ein neues Ticket aus der Mitte und belege dieses sofort mit einer TS.**



## Gesetzesänderung



Es gibt eine Gesetzesänderung, die ab nächstem Monat greift und die eine Änderung am System notwendig macht. Leider hat das Management zu spät von diesem Gesetz erfahren und die Änderung muss jetzt mit höchster Priorität bearbeitet werden.

**Tausche dein aktuelles Ticket gegen ein neues Ticket aus der Mitte und belege dieses sofort mit einer TS.**



## Vernachlässigte Testpflege



Um Zeit und Ressourcen zu sparen, werden Testanforderungen und die Testdatenbank nicht regelmäßig aktualisiert. Dadurch wird das System fehleranfällig und die Anzahl der post-release Bugs steigt, was zu Kundenfrustration führt und den Ruf des Unternehmens schädigen kann.

**Falls du zwei oder mehr TS im System hast, setze eine Runde aus.**



## Verkettete Konsequenzen



Durch eine verpasste Deadline, müssen Entwickler nun Überstunden machen. Ausgelöst von dem daraus resultierendem Stress, versuchen sie einen Schuldigen zu finden. Doch dieses unproduktive Verhalten führt nur zu noch mehr zeitlichen Druck.

**Jedes Ticket in deinem System, das zwei oder mehr TS hat, bekommt eine weitere TS, die dein Gegner platzieren darf.**



## Zeitlicher Druck



Um das aktuelle Ticket fristgerecht abzugeben, muss eine provisorische Lösung implementiert werden.

**Dein Gegner darf eine TS auf dein aktuelles Ticket legen.**



## Innerer Teufelskreis



Entwickler haben sich in der Vergangenheit für eine provisorische Lösung entschieden, die zu unleserlichem Code führte. Daraus entsteht nun ein Teufelskreis, in dem sich Entwickler für weitere provisorische Lösungen entscheiden und mehr unleserlichen Code verursachen.

**Falls du zwei oder mehr TS in deinem System hast, darf dein Gegner eine TS auf ein beliebiges Feature-Ticket legen.**



## Entwickler in Rente



Ein guter Entwickler geht in Rente. Dieser hat zwar guten Code geschrieben, aber leider wenig dokumentiert. Somit müssen die Nachfolger viel Zeit investieren, um seinen Code zu verstehen.

**Du setzt eine Runde aus.**



## Architekturentscheidungen



Du stellst fest wie sehr die Qualität deines Systems von Architekturentscheidungen abhängig ist.

**Falls du drei oder mehr TS im System hast, darf dein Gegner eine TS auf ein beliebiges Architektur-Ticket legen.**