Wir starten in die Projektarbeit

Bild: pexels.com

Unser Plan für die nächsten Wochen

Erste Phase: Planen

Zweite Phase: Implementieren

Themenfindung | ***Planning***

Bis Freitag
(4. Februar)

Bis Freitag
(11. Februar)

Themen & Teams — ***Shared Vision*** — **Erste *User Stories*** — ***Sketching***

Wir stehen während der Projektarbeit als Ansprechpartner zur Verfügung und gebleiten Sie dynamisch durch die Projekte.

Sie implementieren **kontinuierlich neue Features** in Ihre Anwendung

Wir starten **heute** in die Projektarbeit

Mindestens zweimal treffen wir uns zu einer verpflichtenden Besprechung des aktuellen Stands

Ende März veröffentlichen Sie eine coole und vollständige Web-Anwendung

Unser Plan für heute

bis 12:30 Uhr

Einführung und Überblick

Ich stelle Ablauf und Ziele des heutigen Workshops vor.

12:30 – 12:50 Uhr

Einführung **User Stories**

Ich stelle kurz vor, wie gute *User Stories* aufgebaut sind und wie diese verwendet werden können.

12:50 – 14:00 Uhr

User Stories schreiben **Gruppenarbeit**

Wir schauen währenddessen kurz bei jedem Team vorbei ;)

Die Teams versuchen die wichtigsten Anforderungen der jeweiligen Anwendung in Form **guter** und **verwendbarer** *User Stories* festzuhalten.

ab 14:15

Einführung **Sketching**

Ich stelle kurz vor, wie Sie gemeinsam gute Ideen für die Umsetzung wichtiger Bereiche Ihres UIs entwickeln können.

Im Anschluss

Sketching zentraler UI-Elemente **Gruppenarbeit**

Die Teams definieren zentrale UI-Bereiche der Anwendung und versuchen kollaborativ und iterativ Design-Vorschläge für deren Umsetzung zu entwerfen.

ab 15:30 Uhr

Kurzpräsentation (2 Minuten)

Jedes Team stellt die wichtigste *User Story* und die damit verbundene UI-Skizze vor.

In Zoom (Bildschirmübertragung vom eigenen Laptop)

Treten Sie Ihrem Team bei:
classroom.github.com/a/Slm0caX8

Bild: pexels.com

Bild: pexels.com

Bild: pexels.com

User Stories

Defintion: Anforderungen [Sommerville 2012]

- Anforderungen (*Requirements*) beschreiben den Funktionsumfang einer Software und spiegeln die Bedürfnisse der Kund*innen bzw. potenzieller Nutzer*innen im Umgang mit der Software wieder.
- Anforderungen beschreiben was eine Software tut (funktionale Anforderungen) und wie sie sich dabei verhält (nicht-funktionale Anforderungen).
- Anforderungen werden Erhoben, Dokumentiert, Implementiert und Überprüft.
- Der Begriff des *Requirements Engineering* beschreibt alle Prozesse die im Rahmen der Erstellung und Nutzung von Anforderungen existieren.

User Stories != Anforderung [Demuth & Sneed 2016]

- Ein *Backlog* mit *User Stories* ersetzt nicht die Anforderungsdokumentation!
- *User Stories* dienen der Kommunikation von bereits definierten Anforderungen.
- Eine *Story* beschreibt eine einzelne Anforderung und erlaubt die Diskussion über und die Integration von dem entsprechenden Feature.
- Eine *User Story* wird im Kontext bekannter Anforderungen erstellt.

User Stories in Softwareprojekten [Cohn 2004]

Die Anforderungen und Features eines Softwareprojekts können in Form von *User Stories* erhoben und/oder dokumentiert werden:

- Jede User Story bildet kleinteilig einen elementaren Bestandteil des gesamten Produkts ab.
- Jede User Story schildert eine in sich unabhängige direkte oder indirekte Interaktion der Nutzer*innen mit dem Produkt.
- Jede User Story stellt – wenn implementiert – einen messbaren und singulären Mehrwert für das Produkt dar.
- Jede User Story sollte möglichst unabhängig von anderen Aspekten implementiert werden können.
- Jede User Story kann einem übergeordneten *Epic/Theme*, einer größeren, zusammenhängenden Story, zugeordnet werden.

User Stories in Softwareprojekten [Jeffries 2001]

User Stories lassen sich hinsichtlich ihrer Bedeutung innerhalb eines Softwareprojekts anhand dreier Aspekte (3Cs) beschreiben:

Card | Die eigentliche (physikalische) Formulierung der *User Story.* Die Karte enthält nicht alle anforderungsspezifischen Informationen. Sie repräsentiert die Anforderung und erlaubt Entwickler*innen diese zu identifizieren.

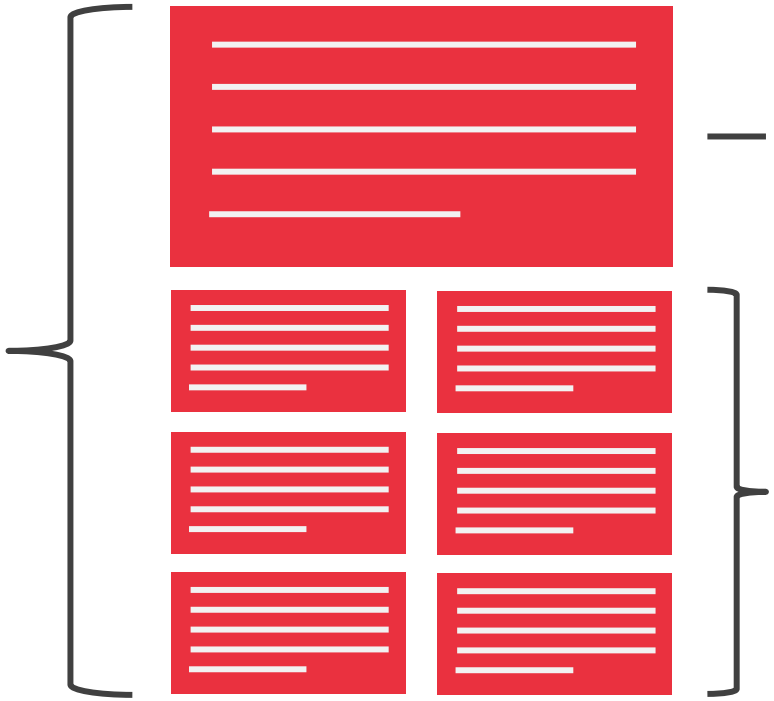
Conversation | Eine Anforderung entsteht im Rahmen einer Diskussion. Der Inhalt einer entsprechenden *User Story* wird im Laufe des Projekts diskutiert und angepasst. Die Diskussion sollte im Kontext der Karte festgehalten werden.

Confirmation | Die Umsetzung jede *User Story* muss abschließend beurteilt werden können. Ob eine *Story* tatsächlich vollständig und zufriedenstellen implementiert wurde, muss durch die Projektbeteiligten und definierte Akzeptanzkriterien festgestellt werden können.

User Stories, Epics und Themes [Cohn, 2011]

User Stories lassen sich hinsichtlich ihres Umfangs kategorisieren und bündeln. Die Begriffe *Epic* und *Themes* sind dabei wenig trennscharf und können auch anders interpretiert werden:

Thematisch zusammenhängende *Stories* können unter einem ***Theme*** zusammengefasst werden.



Umfassendere *Stories* die bewusst größere Bereiche des Produktes betreffen oder noch nicht in kleinteiligere *Stories* herunter gebrochen wurden, werden als ***Epics*** bezeichnet.

Einzelne ***User Stories*** bilden elementare, alleinstehende Features bzw. Anforderungen ab.

Beispiele für: Story, Epic und Theme

Kontext: Das fiktives CMS-System einer Website

Theme: Benutzerprofil

Nutzer*innen verfügen über ein persönliches Profilkonto, das über einen Nutzername identifiziert und mit einem Passwort gesichert ist. Das Profil umfasst einen Nutzernamen, eine Email-Adresse und ein Profilfoto.

Epic: Benutzerprofil verwalten

Als Nutzer*in möchte ich die in meinem Profil gespeicherten Informationen einsehen und editieren können um die gespeicherten Werte aktualisieren zu können.

User Story: Passwort ändern

Als angemeldete Nutzer*in möchte ich in meinem Benutzerprofil mein persönliches Passwort ändern können.

User Story: Passwort zurücksetzen

Als Nutzer*in möchte ich mein persönliches Passwort zurücksetzten können, wenn ich meine Anmeldeinformationen vergessen habe.

Das INVEST-Schema [Wake 2003]

Eine User Story ist von hoher Qualität, wenn sie bestimmte Kriterien erfüllt, die einen effektiven Einsatz im Rahmen der Projektentwicklung erlauben. Eine Möglichkeit zum Erstellen und Bewerten von Stories ist das INVEST-Schema:

I **Independent** | Eine *User Story* sollte möglichst unabhängig von anderen *Stories* formuliert sein. Das ist nicht immer umsetzbar, sollte aber hinsichtlich der Plan- und Steuerbarkeit der Umsetzung angestrebt werden.

N **Negotiable** | Eine *User Story* ist kein unveränderbarer Vertrag. Sie wird gemeinsam von Kund*innen und Entwickler*innen erstellt. Eine gute *Story* erfasst die wichtigsten Aspekte eines Features, gibt aber keine Details zur Implementierung vor. Eine gute *User Story* wird im Laufe des Projekts weiterentwickelt.

V **Valuable** | Eine *User Story* muss einen messbaren Wert für Kund*innen bzw. Nutzer*innen haben. Sie beschreibt ein tatsächliches Feature und keine *nicht sichtbaren* Aspekte, wie z.B. Datenbanken oder Authentifizierungsvorgänge.

E **Estimable** | Eine *User Story* ist hinsichtlich des Zeitaufwands zur Implementierung abschätzbar. Eine exakte Einschätzung ist dabei nicht nötig. Die *Stories* sollten aber untereinander vergleich- und priorisierbar sein.

S **Small** | Eine *User Story* ist möglichst kleinteilig formuliert. Sie beschreibt ein Feature, dass in einer überschaubaren Zeit umgesetzt werden kann.

T **Testable** | Eine *User Story* ist so formuliert, dass ein überprüfbares Ziel klar erkennbar ist. Es muss klar definierbar sein, ob eine *Story* vollständig implementiert worden ist.

Ein Template für gute User Stories [Cohn 2008]

- Eine einheitliche Struktur vermindert Qualitätsunterschiede innerhalb des *Backlogs*.
- Ähnlich formulierte *Stories* erlauben eine schnelle Identifikation und ein schnelles Verstehen der wichtigsten Punkte.
- Ein *Template* stellt sicher, dass die wichtigsten Bestandteile und Ziele einer Story berücksichtigt werden.
- (Nutzer-zentriert formulierte *Stories* erzeugen eine bessere Nachvollziehbarkeit der geschilderten Probleme.)

Role, Feature & Benefit

Ursprünglich von einem Team Firma *Connextra* um Tim Mackinnon entwickelt.

Als ein <Rolle der NutzerInnen> möchte ich <Ziel> um <Begründung>.

Rolle: Die *Story* wird aus der Perspektive konkreter Nutzer*innen mit einer bestimmten Aufgabe geschildert. Im Rahmen der Anforderungsanalyse kann diese Perspektive auch durch den Einsatz von *Personas* gewonnen werden.

Ziel: Das Ziel der Interaktion der Nutzer*in mit dem System wird präzise formuliert und sollte in einen möglichst kleinteiligen Kontext eingebettet sein.

Begründung: Jede *User Story* sollte die Intention von Nutzer*innen klar wiedergeben. Der Sinn und Zweck hinter dem beschriebenen Ziel muss deutlich hervorgehoben werden. Dieser Aspekt einer *User Story* ist möglicherweise optional (Vgl.: [Cohn 2008]).

Role, Feature & Benefit: Beispiele

#1 | Als **Administrator*in der Website** möchte ich **Benutzergruppen erstellen** können, um **existierende Nutzer in diesen organisieren zu können**.

#2 | Als **Administrator*in der Website** möchte ich **Benutzergruppen spezifische Rechte zuweisen**, um den **Zugriff auf bestimmte Teilbereiche der Website zu steuern**.

#3 | Als **Autor*in der Website** möchte ich **eine Liste aller von mir editierbaren Teilbereichen einsehen**.

MME Live präsentiert den **Fortschritt** der laufenden Softwareprojekt des *Multimedia Engineering*-Kurses als übersichtliches **Dashboard** für alle Teilnehmer*innen des Kurses.

Das Problem: Im Rahmen der Abschlussprojekte arbeiten mehrere Teams nebeneinander an unterschiedlichen Projekten mit technisch sehr ähnlichen Hintergründen. Ein direkter Austausch oder Vergleich mit anderen Teams kann dabei konstruktives Feedback für das eigene Vorgehen mit sich bringen und die einzelnen Teams für das weitere Vorgehen motivieren. Als Ausgangslage für diese Art der Zusammenarbeit fehlt eine zentrale Plattform, die die verschiedenen Projekte vorstellt und den individuellen Fortschritt bzw. den aktuellen Stand der jeweiligen Anwendungen präsentiert.

Die Lösung: Wir implementieren ein interaktives Web-Dashboard, das auf Basis der öffentlich einsehbaren Informationen aus den GitHub-Repositories der einzelnen Teams den aktuellen Stand der Projekte aufbereitet visualisier. Im Dashboard wird anhand automatisch generierter Screenshots die aktuelle Version der Projekte präsentiert und die Dynamik bzw. der Fortschritt der Projektarbeit auf Basis der Repository- und Issue-Tracker-Aktivitäten visualisiert. Nutzer*innen können die aktuellen Fassung der Projekte ausprobieren, besondere Ereignisse bewerten und den anderen Team konstruktives Feedback zu kommen lassen.

Tim,
Student

*„Um den Fortschritt meines eigenen Teams besser einschätzen zu können, möchte ich mir schnell und einfach einen Überblick darüber verschaffen, wo die anderen Teams aktuell stehen. Ich möchte die anderen Projekte ausprobieren können um Ideen für unsere Anwendung zu sammeln und um meine Kommiliton*innen mit Feedback zu unterstützen.“*

Anna,
Dozentin

*„Ich möchte auf einen Blick sehen, wie weit meine Student*innen schon mit Ihren Projekten vorangeschritten sind. Dabei möchte ich positive Entwicklungen schnell bemerken und bestätigen, aber auch mögliche Probleme möglichst leicht identifizieren können. Um die Team bestmöglich zu unterstützen, möchte ich schnell auf die veröffentlichen Versionen der Projekte und die wichtigsten Bereiche der Repositories zugreifen können.*

Alle Bilder: pexels.com

Werte und Ziele

Wir liefern Übersichtlichkeit · Informativität · Aktualität

Wir visualisieren Fortschritt · Dynamik · Erfolge

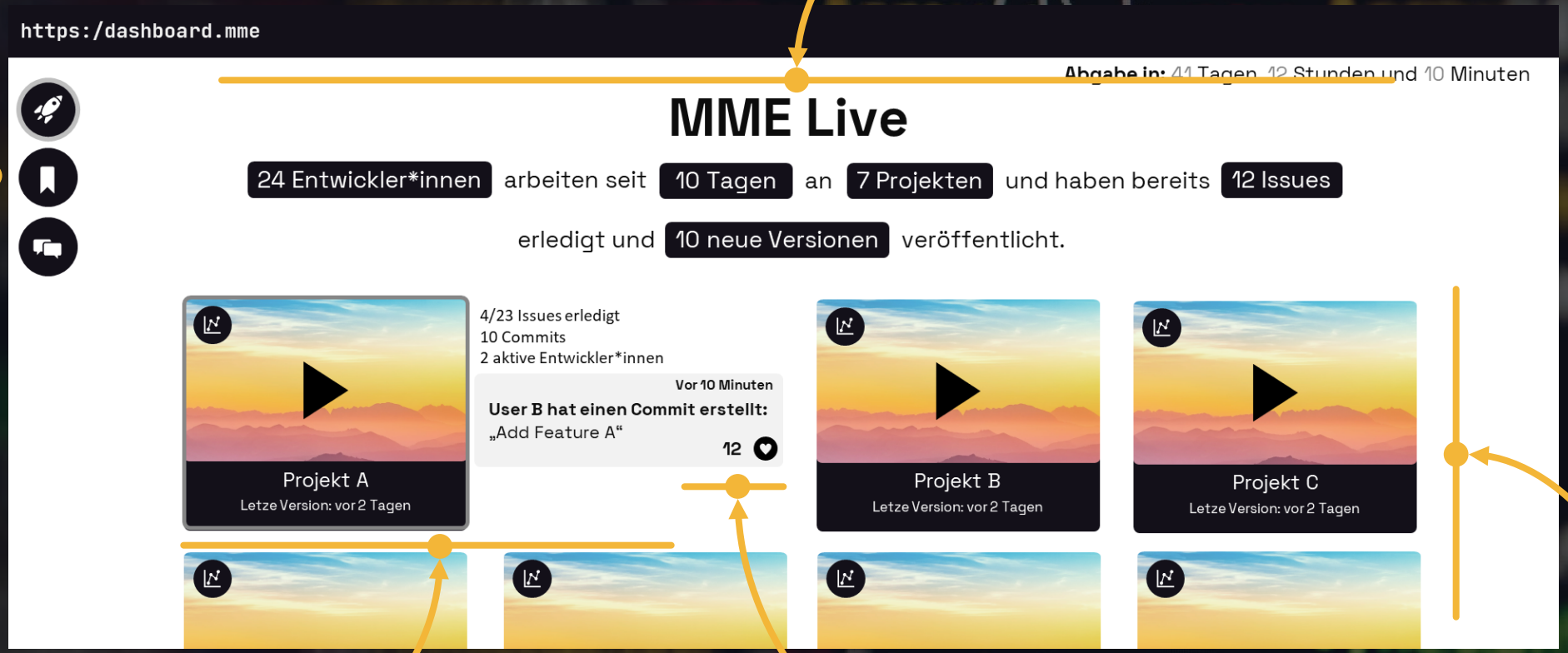
Wir ermöglichen Feedback · Inspiration

Alle Bilder: pexels.com

User Interface: Zentrale Ideen

Navigation zwischen den zentralen Bereichen der Anwendung

Dashboard mit automatisch aggregierten Statistiken zu allen Projekten



Detail-Informationen zu aktuell ausgewählten Projekten

Direkt Integrierte **Interaktionsmöglichkeiten**

Durchsuchbare **Liste** aller Projekte als Informations-**Widgets**

Alle Bilder: pexels.com

Ein Beispiel

Für die Umsetzung der Story benötigen wir wahrscheinlich weitere Infrastrukturen (z.B. eine eigene API zum Erzeugen der Screenshots). **Die Arbeit daran ist Teil der Story ist impliziter Teil der Story und muss in deren Kontext umgesetzt bzw. diskutiert werden**.

„Vorderseite“

User Story: Dashboard-Überblick

Als Nutzer*in möchte ich einen **vollständigen Überblick** über alle aktuell bearbeiteten MME-Projekte gewinnen um mir **schnell ein Gesamtbild machen zu können**. Dazu möchte ich direkt sehen können, wie die **aktuelle Version** der jeweiligen Projekte **aussieht** und wann die Teams die Projekte **zuletzt bearbeitet haben**.

Ein „und“ kann, muss aber nicht, Indikator für eine nicht vollständig heruntergebrochene Story sein.

„Rückseite“

- **Wir müssen die notwendigen Informationen zu allen Projekten *live* beziehen.**
- **Wir müssen automatisiert Screenshots der aktuellen Versionen erstellen.**
- **Wir müssen die Informationen übersichtlich am Bildschirm anordnen.**

Mit dieser Story beginnen wir die Umsetzung. Sie enthält alles, was für die Nutzer*innen wichtig ist. Implementierungsdetails erarbeiten wir uns während der Umsetzung (Rückseite).

Vorgehen

Sprechen Sie miteinander und arbeiten Sie gemeinsam. Ziel ist nicht, dass jedes Teammitglied N Stories in den *Issue Tracker* einträgt, sondern dass Sie gemeinsam gute Stories für die weitere Arbeit an dem Projekt schreiben.

Halten Sie Ideen direkt fest. Beginnen Sie direkt im *Issue Tracker* und schreiben Sie jeden Gedanken auf. Sie können alle *Stories* jederzeit ändern, ersetzen, verbessern, erweitern oder aufteilen.

Sammeln Sie Ideen und Stories. Beginnen Sie mit groben Beschreibungen und decken Sie möglichst viele Bereiche der Anforderungen/Anwendung ab.

Zerlegen Sie größere Stories. Die initialen *Stories* werden mit Sicherheit nicht kleinteilig genug sein, um diese für die Umsetzung der Anwendung zu verwenden. Teilen Sie die *Stories* so lange auf, bis handhabbare Aufgaben entstehen.

Verbessern Sie kontinuierliche alle Stories. Wenn Ihnen bessere Formulierungen einfallen: Scheuen Sie sich nicht, bereits bestehende Stories erneut anzufassen und zu verbessern!

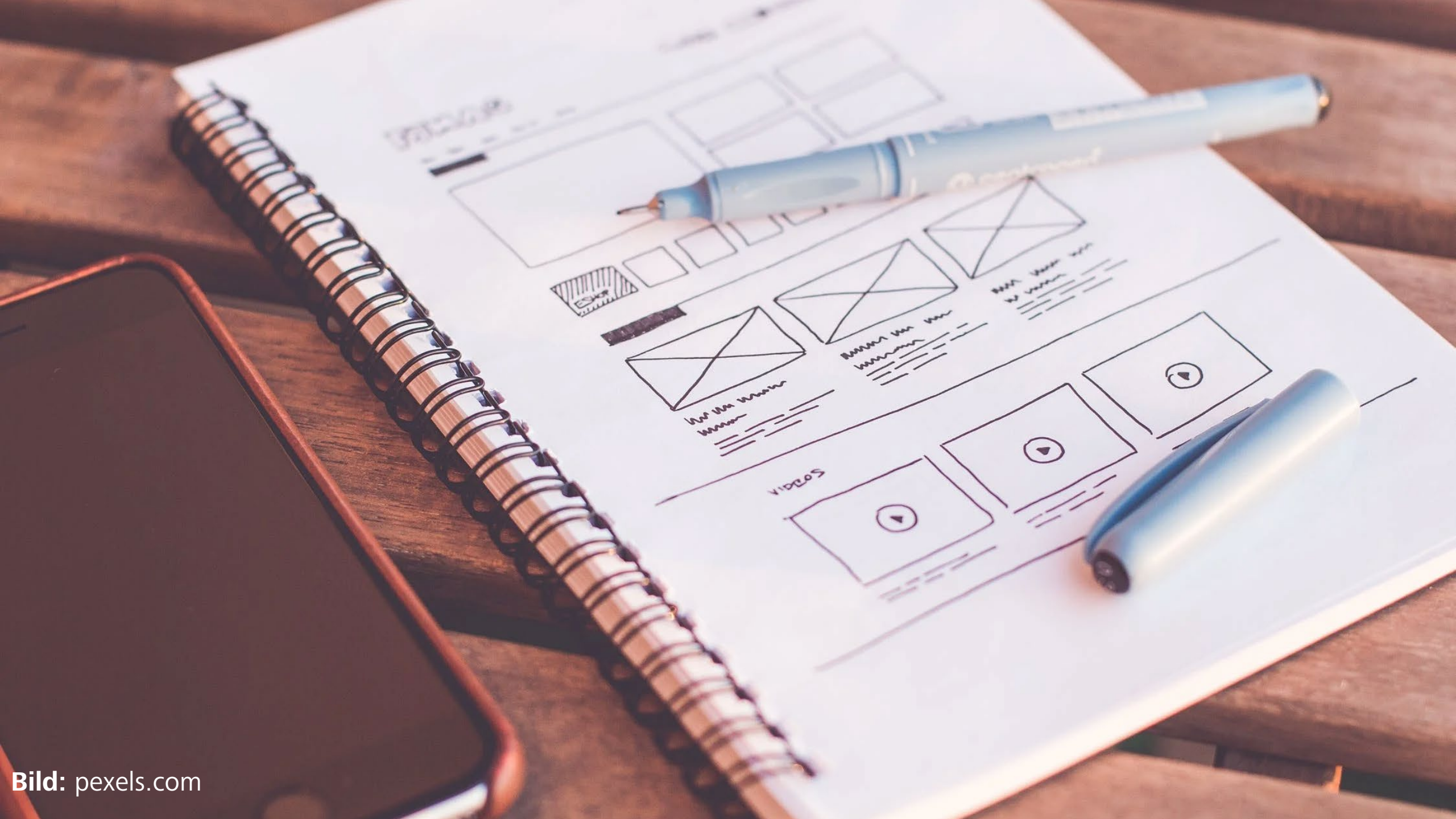


Bild: pexels.com

Kollaboratives, iteratives Sketching

Wählen Sie einen zentralen Bereich des UIs aus (ein *Screen*)

Alle Teammitglieder entwerfen einen eigenen Vorschlag (Skizze)

Setzen Sie ein Zeitlimit (z.B. 5 Minuten)

Verwenden Sie Stift und Papier

Stellen Sie die Vorschläge vor und diskutieren Sie die Lösungsansätze

Wählen Sie den „besten“ Vorschlag aus

Alle Teammitglieder entwerfen auf Basis des Vorschlags eine verbesserte Version

Wiederholen Sie den Vorgang, bis es keine signifikanten Verbesserungs-vorschläge mehr gibt.

Annotieren Sie Ihre Skizzen

Heben Sie wichtige Elemente hervor, statt unwichtige Linien auszuradieren

Verwenden Sie **Platzhalter** für Elemente, die für die aktuelle Skizze nicht wichtig sind

Skizzieren Sie **Details** in **separaten Elementen**