Karlsruher Institut für Technologie

Klausur

Softwaretechnik II

Sommersemester 2024

Prof. Dr. Ralf H. Reussner

13.09.2024

Name: _____

Matrikelnummer: _____

Zur Klausur sind keine Hilfsmittel und kein eigenes Papier zugelassen. Die Bearbeitungszeit beträgt 90 Minuten. Die Klausur ist vollständig und geheftet abzugeben. Verwenden Sie ausschließlich dokumentenechte Stifte. Mit Bleistift oder roter Farbe geschriebene Angaben werden nicht bewertet. Lösen Sie die Aufgaben in leserlicher Schrift. Unlesbare Lösungen können naturgemäß nicht positiv bewertet werden. Lesen Sie die Aufgabenstellungen ordentlich und geben Sie nicht mehr Lösungen als gefordert an.

Wenn das generische Maskulinum gewählt wurde, geschieht dies zur besseren Lesbarkeit und zum einfachen Verständnis der Aufgabenstellung. Sofern nicht anders angegeben, beziehen sich Angaben im Sinne der Gleichbehandlung auf Vertretende aller Geschlechter.

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	Σ
Maximal	10	19	18	13	22	8	90
K1							
K2							
K3							

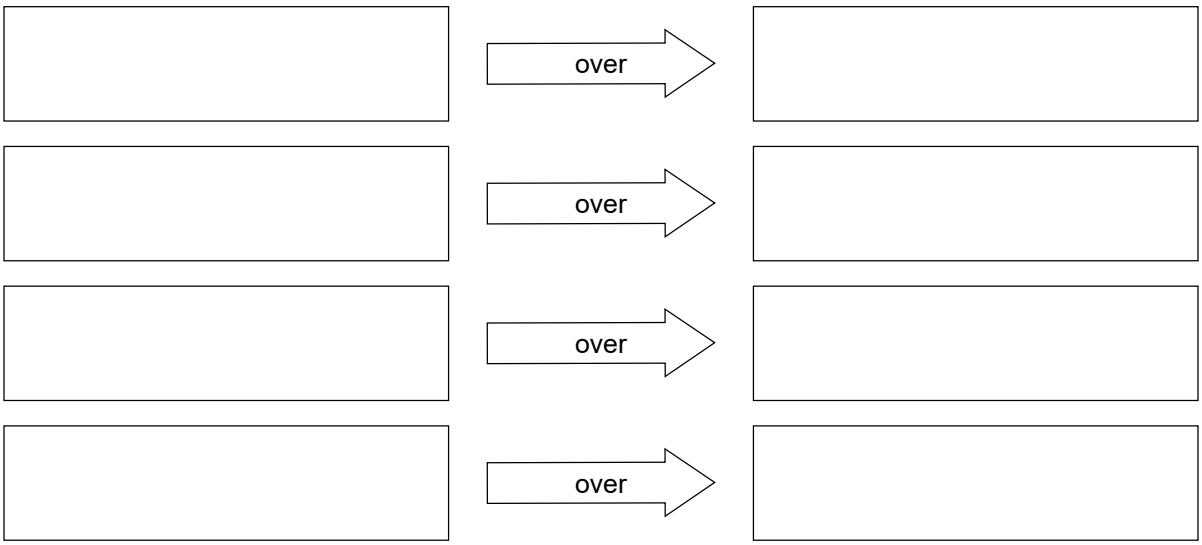
Aufgabe 1: Verschiedenes (10 Punkte)

a) Was ist eine Komponente und was zeichnet sie aus? (1 Punkt)

b) Sie haben in der Vorlesung die Muster *Transaction Script*, *Domain Model* und *Table Module* zur Repräsentation von Domänenlogik kennengelernt. Beschreiben Sie jeweils in einem Satz, was das jeweilige Muster auszeichnet und nennen Sie jeweils einen Vor- und Nachteil. (6 Punkte)

c) Was meint man mit der Skalierbarkeit (*scalability*) und Elastizität (*elasticity*) eines Softwaresystems? Erklären Sie die beiden Begriffe kurz. (1 Punkt)

d) Füllen Sie in die folgenden Kästen die Aussagen des Agilen Manifests (*Agile Manifesto*) ein. (2 Punkte)



Aufgabe 2: Anforderungen (19 Punkte)

Ihr Unternehmen bekommt den Auftrag, die Plattform *OnlyPhiit* zu entwickeln. Hierfür sollen Sie zunächst die Anforderungen erheben und spezifizieren. Sie erhalten dafür folgendes Schreiben:

> Liebe SWT2-AG,
> wir freuen uns, dass Sie die Entwicklung unserer Plattform *OnlyPhiit* übernehmen. *OnlyPhiit* soll eine Plattform für Künstler und Kreative sein, die ihre Werke verkaufen und mit anderen Nutzern teilen möchten. Die Plattform soll es den Nutzern ermöglichen, sich zu vernetzen und gegenseitig zu unterstützen. Kunden können bestimmte Einsteller von Inhalten abonnieren, deren Werke anschließend in ihrem persönlichen Feed erscheinen. Kunden und Einsteller von Inhalten erhalten zudem die Möglichkeit sich auszutauschen. Dies geht einerseits mittels persönlicher Nachrichten oder in LiveChats neben den eingestellten Inhalten. Um die Plattform für Künstler attraktiv zu machen, bieten wir Künstlern verschiedene Bezahlmodelle an, sodass sie ihre Werke als Abo oder einzeln anbieten können. Die interne Bezahlung erfolgt dann über Tokens.

a) Vergleichen Sie verschiedene Techniken zur Anforderungserhebung bezüglich ihrer generellen Stärken und Schwächen. Markieren Sie hierzu in der folgenden Tabelle für jede Kategorie, ob die Technik im Gegensatz zu anderen Techniken stark (+), schwach (-) oder neutral (o) ist. (12 Punkte)

Technik	**Bedürfnisse ausdrücken**	**Möglichkeiten demonstrieren**	**Aktuelles System analysieren**	**Marktpotential explorieren**
Interviews				
Fragebögen/ Umfragen				
Prototypen/ Mock-ups				
Rollenspiele				
Stakeholder-Beobachtung				
Marktstudien				

b) Sie entscheiden sich dafür einen Fragebogen an die Stakeholder von *OnlyPhiit* zu senden. Definieren Sie den Begriff Stakeholder und nennen Sie ein Beispiel, welches nicht oben im Text genannt wurde. (1 Punkt)

c) Aus ihrer Umfrage erhalten Sie folgende Anforderungen. Klassifizieren Sie die Anforderungen nach der in der Vorlesung behandelten Klassifikation von Glinz. Ordnen Sie hierzu die vier Anforderungen den Anforderungsarten (*kind*), deren Unterkategorie (*subkind*) sowie deren Repräsentation (*representation*) zu. (6 Punkte)

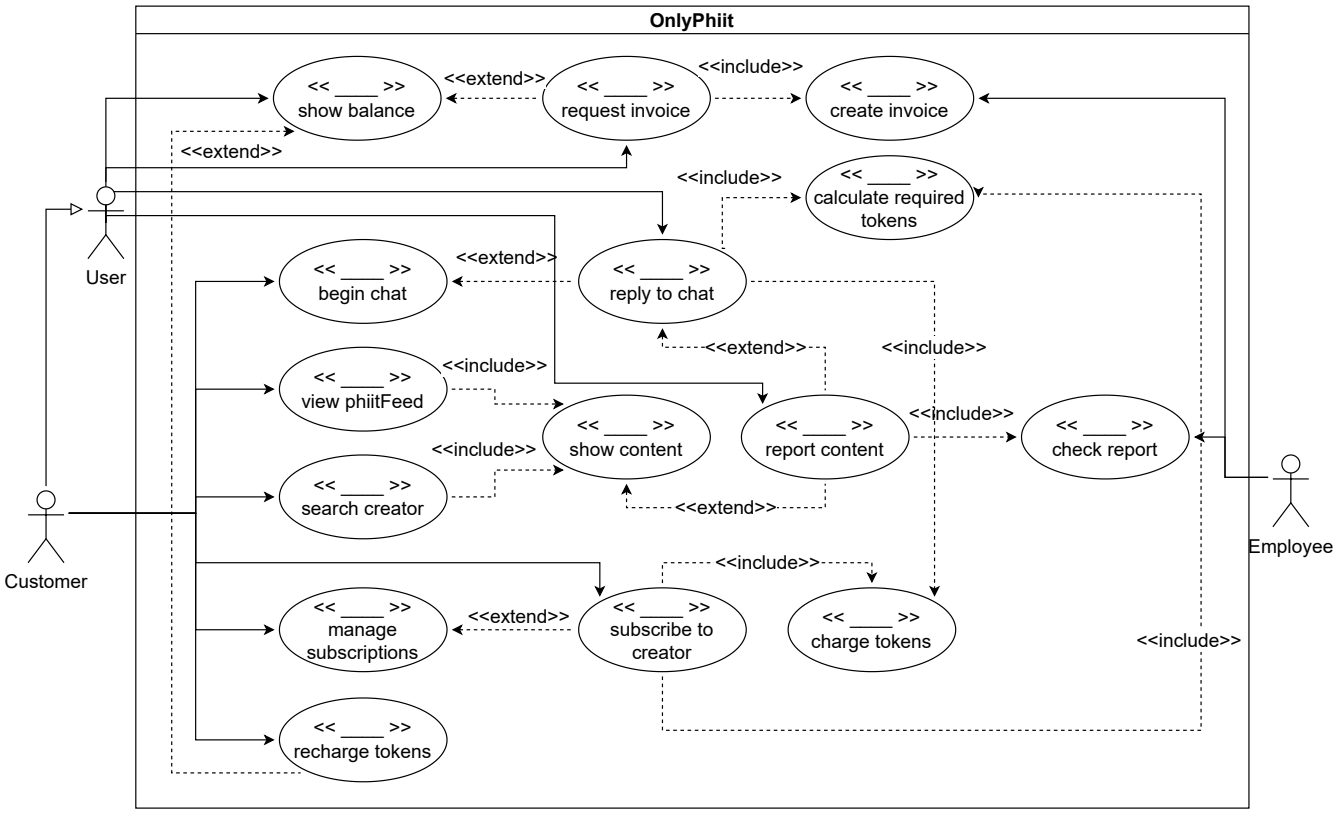
Nr.	**Anforderungsbeschreibung**
R1	Der persönliche Feed soll von jeder *OnlyPhiit*-Ansicht mit maximal einem Klick erreichbar sein.
R2	Wenn ein Einsteller ein Bild hochlädt, soll dieses auf unangemessene Inhalte geprüft werden, bevor es öffentlich gemacht wird.
R3	Bilder sollen innerhalb von 0.5 Sekunden geladen werden.
R4	Die Plattform soll auf iOS und Android verfügbar sein.

Nr.	**Art**	**Unterkategorie**	**Repräsentation**
R1			
R2			
R3			
R4			

Aufgabe 3: Anwendungsfälle (18 Punkte)

Basierend auf den Anforderungen aus Aufgabe 2 wird ein Anwendungsfall für einen Prototypen von *OnlyPhiit* erstellt.

a) Klassifizieren Sie jeden Anwendungsfall im folgenden Anwendungsfalldiagram (*use case diagram*) entweder als Nutzerziel (*user goal*) mit U, als Teilfunktion (*subfunction*) mit S oder als Operation (*operation*) mit O. Tragen Sie hierzu die Klassifikation zwischen den Klammern «____» ein. (7,5 Punkte)

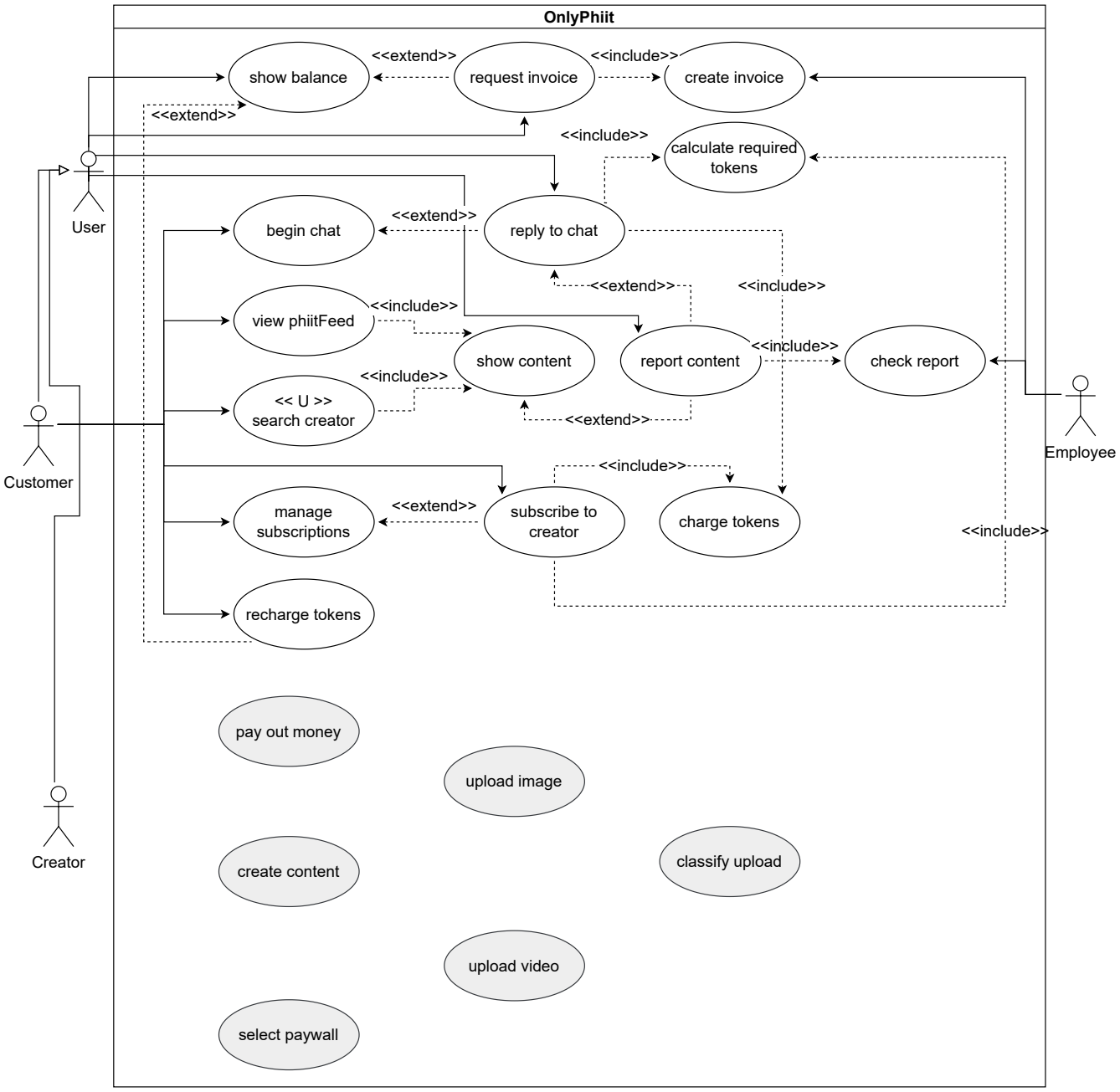


Anwendungsfall	**Beschreibung**
show balance	Anzeigen des aktuellen Guthabens in Token
request invoice	Erfragen der Zahlungshistorie
create invoice	Erstellen einer Rechnung durch einen Mitarbeiter basierend auf der Zahlungshistorie eines Kunden
begin chat	Starten eines persönlichen Chats
reply to chat	Senden einer persönlichen Chatnachricht
calculate required tokens	Berechnen der benötigten Tokens
view phiitFeed	Anzeigen des persönlichen phiitFeed
search creator	Suchen eines bestimmten Einstellers mit Benutzernamen
show content	Laden und anzeigen von einzelnen Inhalten und Beiträgen
report content	Melden eines unangemessenen Inhalts durch einen Nutzer
check report	Prüfen eines gemeldeten Inhalts durch einen Mitarbeiter
manage subscriptions	Verwalten der persönlichen Abonnements
subscribe to creator	Abonnieren eines Einstellers
charge tokens	Verrechnen der Tokens
recharge tokens	Aufladen der Tokens, die in einem Account hinterlegt sind

Sie schicken das obige Anwendungsfalldiagramm an die Auftraggeber von *OnlyPhiit*. Daraufhin erhalten Sie folgende Rückmeldung:

> Im von Ihnen geschickten Anwendungsfalldiagramm fehlen noch einige Anwendungsfälle für die Einsteller (*Creators*). Einsteller können Inhalte erstellen (*create content*), wobei sie entweder ein Bild oder ein Video hochladen können (*upload image*/ *upload video*). Beim Hochladen von Bildern und Videos wird der Inhalt mittels einer Bilderkennung klassifiziert und auf Unangemessenheit überprüft (*classify upload*). Falls eine potenzielle Unangemessenheit erkannt wird, wird der Inhalt maschinell gemeldet. Hierzu wird intern die gleiche Funktion genutzt, die auch von Nutzern zum Melden von Inhalten verwendet wird (*report content*). Außerdem müssen Einsteller beim Erstellen eines Inhalts ein Bezahlmodell für den Inhalt auswählen (*select paywall*). Das dadurch erworbene Guthaben kann über eine Auszahlfunktion abgebucht werden (*pay out money*). Selbstverständlich können sich Einsteller beim Auszahlen ihrer erworbenen Tokens auch nochmal ihren aktuellen Kontostand anzeigen lassen (*show balance*). Beim Auszahlen der Tokens werden die abzubuchenden Tokens dann mit dem aktuellen Guthaben verrechnet (*charge tokens*).

b) Vervollständigen Sie nun das folgende Anwendungsfalldiagramm (*use case diagram*) genau so, dass es die Anforderungen im obigen Text beschreibt. Zeichnen Sie ausschließlich die Beziehungen von Aktoren (*actors*) zu Anwendungsfällen und zwischen Anwendungsfällen ein, die im Text beschrieben werden. Zeichnen Sie keine weiteren Anwendungsfälle oder Aktoren ein. (6,5 Punkte)



c) Die folgende Anwendungsfallbeschreibung entspricht nicht dem Fully Dressed Use Case Schema. Erklären Sie warum und beschreiben Sie, was geändert werden müsste, damit die Beschreibung das Schema erfüllt. (4 Punkte)

Hinweis: Beziehen Sie hierzu nur Informationen ein, die für das Anwendungsszenario beschrieben wurden. Die Codierung für die Erweiterungen und Alternativen ist wie folgt: Alternative Abläufe werden durch

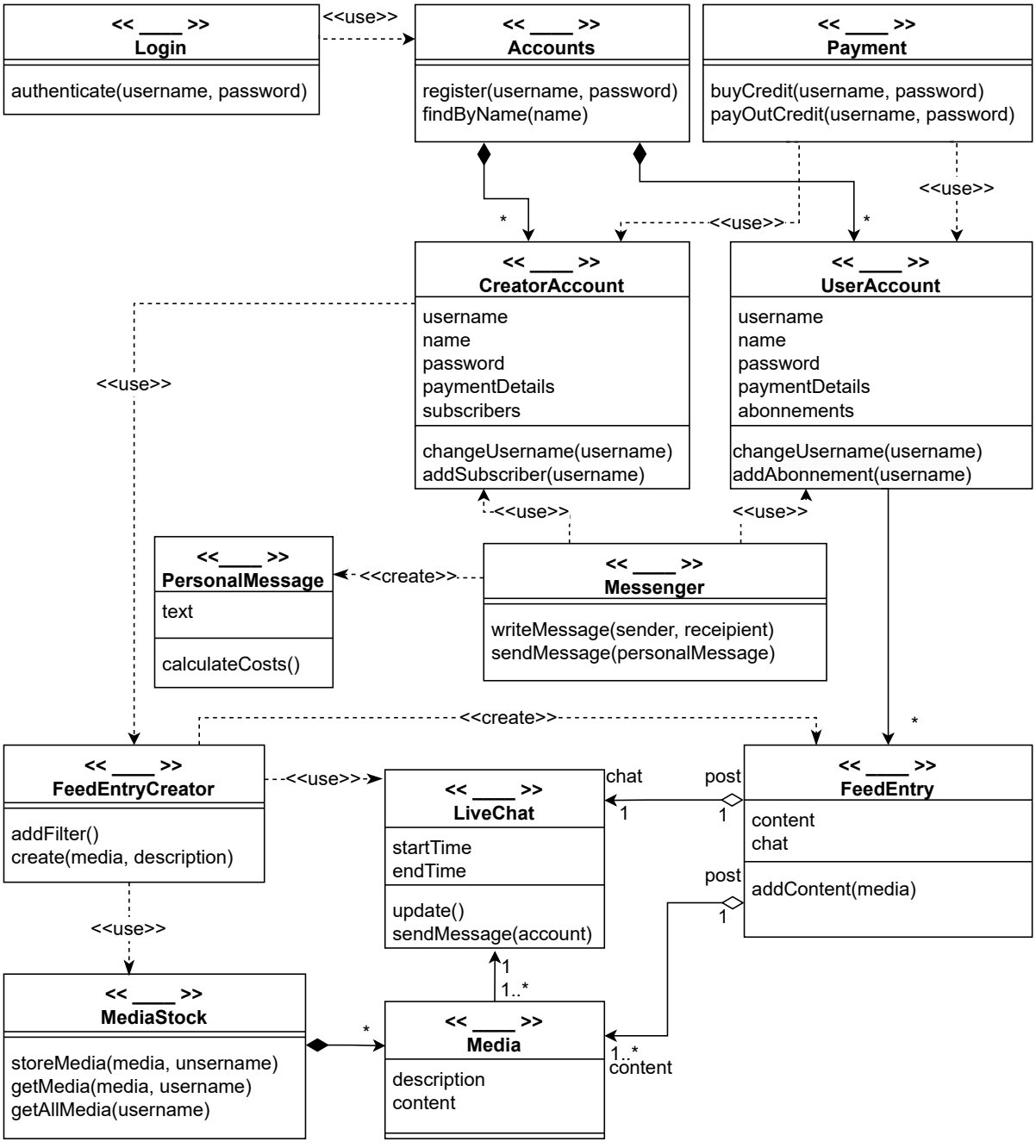
Buchstaben gekennzeichnet (z.B. 3b als Alternative zu Schritt 3 im Erfolgszenario). Erweiterungen werden durch Zahlen gekennzeichnet.

Name des Anwendungsfalls *Use Case Name*	**Einstellen eines Inhalts**
Zielebene *Goal Level*	Kundenziel (Customer Goal)
Primäraktor(en) *Primary Actors*	Einsteller
Stakeholders	Einsteller, Mitarbeiter
Vorbedingungen *Preconditions*	OnlyPhiit ist online
Nachbedingungen *Postconditions*	Eingestellter Inhalt ist verfügbar
Primäres Erfolgszenario *Main Success Scenario*	1. Einsteller wählt aus, dass er Inhalte erstellen möchte 2. Ein vom Einsteller ausgewähltes Bild wird hochgeladen 3. Das Bild wird als nicht gefährdend klassifiziert
Erweiterung (Weiterer Ablauf) *Extension; Additional Flow*	3b1 Das Bild wird als gefährdend eingestuft 3b2 Es wird ein Bericht über das Bild erstellt 3b3a Der Bericht wird durch einen Mitarbeiter geprüft und freigegeben 3b3b Der Bericht wird durch einen Mitarbeiter geprüft und nicht freigegeben
Spezielle Anforderungen *Special Requirements*	usability design constraints
Technologie und Datenvariationen *Technology and Data variations*	-

Aufgabe 4: Domänengetriebener Entwurf (13 Punkte)

a) Erklären Sie die Bausteine des Domänengetriebenen Entwurfs: `ValueObject`, `Entity`, `Service`, `Aggregate`, `Factory`, `Repository` und `Module`. Nennen Sie hierbei wichtige Eigenschaften und Besonderheiten der Bausteine. (7 Punkte)

b) Durch die Domänenanalyse des von *OnlyPhiit* ist das folgende Klassendiagramm entstanden. Klassifizieren Sie jedes Domänenkonzept möglichst spezifisch entsprechend den Bausteinen (building blocks) des Domänen-getriebenen Entwurfs entweder als `ValueObject` mit `VO`, `Entity` mit `E`, `Service` mit `S`, `Factory` mit `F`, `Repository` mit `R` oder `Aggregate` mit `A`. Tragen Sie hierzu die Klassifikation zwischen den Klammern «____» ein. (6 Punkte)



Aufgabe 5: Saubere Unternehmensarchitekturen (22 Punkte)

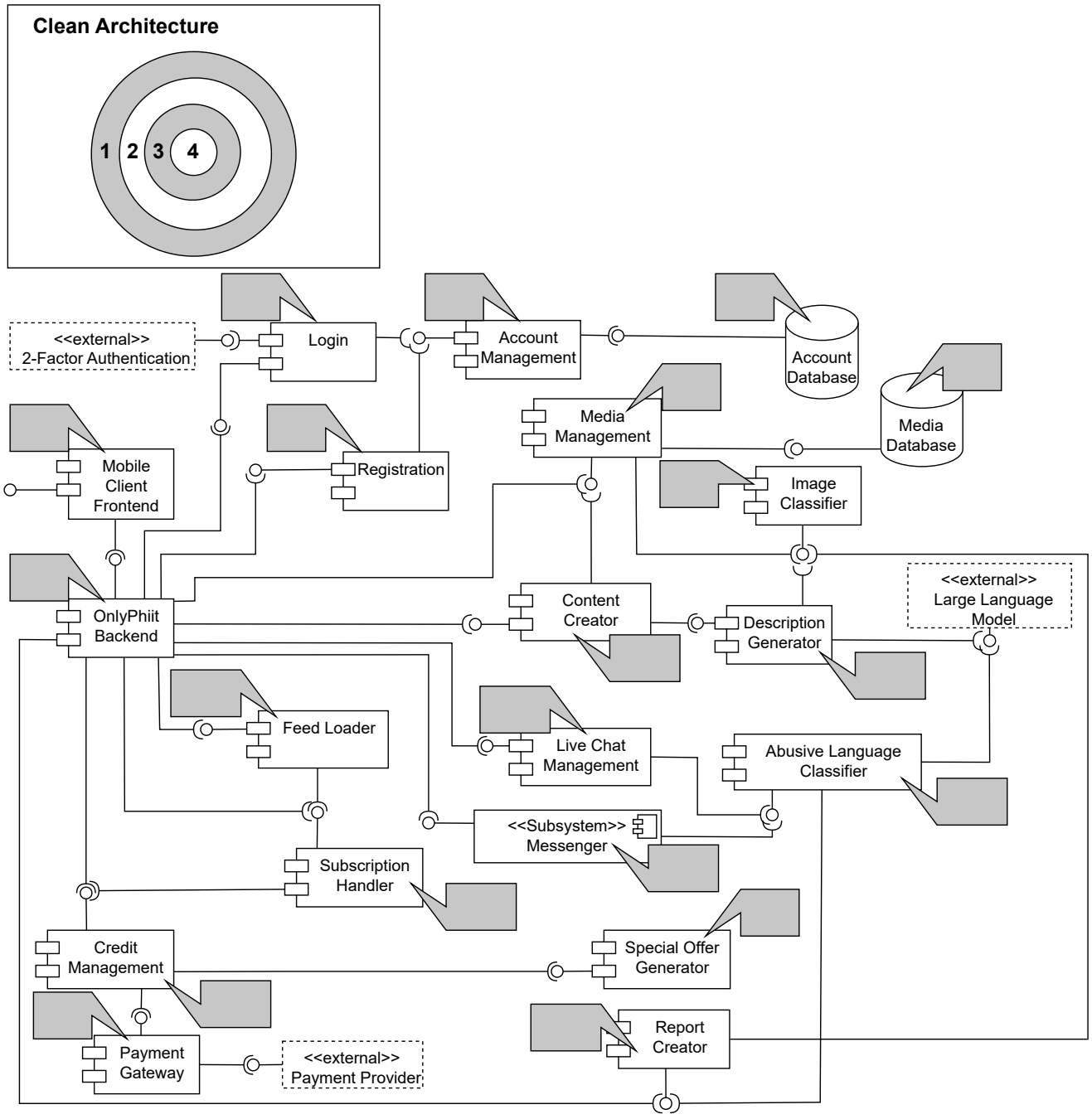
a) Welche Regel wird durch die saubere Architektur (*Clean Architecture*) sichergestellt und wozu dient sie? (1 Punkt)

b) Was kann man tun, wenn eine Architektur gegen die Eigenschaften einer sauberen Architektur (*Clean Architecture*) verstößt? Nennen und erklären Sie das anzuwendende Prinzip. (1 Punkt)

c) Das Komponentendiagramm auf der nächsten Seite stellt die Softwarearchitektur von *OnlyPhiit* dar. Ordnen Sie jede Komponente einer Schicht der sauberen Architektur (*Clean Architecture*) zu, indem Sie die Nummer der Schicht in die Kästen eintragen. Beachten Sie dabei die folgenden Hinweise, die Sie von Ihrem Auftraggeber erhalten: (10 Punkte)

> Unser Unternehmen betreibt bereits eine große Anzahl ähnlicher Systeme. Diese Systeme unterstützen, wie *OnlyPhiit*, den Austausch persönlicher Nachrichten und das manuelle Anzeigen von Verstößen gegen unsere Richtlinien. *OnlyPhiit* unterscheidet sich von unseren anderen Systemen durch den Newsfeed und das Abosystem. Aufgrund der Inhalte und LiveChats gibt es eine maschinelle Erkennung von Missbräuchen. Um den Nutzern den Wechsel zwischen den Systemen zu erleichtern, soll es eine übergreifende Nutzerverwaltung geben.

d) Markieren Sie im gegebenen Komponentendiagramm vier Verstöße gegen eine saubere Architektur (*Clean Architecture*), indem Sie die Konnektoren durchstreichen. (2 Punkte)



e) Kreuzen Sie an, ob in dem Codeausschnitt auf den nächsten Seiten die folgenden Verstöße gegen die Praktiken des sauberen Programmierens (*Clean Code*) vorliegen oder nicht. Falls ja, nennen Sie die Zeile und erklären Sie kurz, warum in diesem Fall ein solcher Verstoß vorliegt. (8 Punkte)

Hinweis: Gehen Sie davon aus, dass jede Klasse einen Konstruktor besitzt, der alle Attribute der Klasse entgegennimmt und setzt. Gehen Sie bei Ihrer Erklärung spezifisch auf den Code ein und vermeiden Sie zu allgemeine Aussagen.

Verstoß	**Antwort**	**Zeile**	**Begründung**
Open Closed Principle	☐Ja ☐Nein		
Law of Demeter	☐Ja ☐Nein		
Principle of Least Surprise	☐Ja ☐Nein		
Code Conventions: Naming	☐Ja ☐Nein		
Dependency Inversion Principle	☐Ja ☐Nein		
Code Conventions: Commenting	☐Ja ☐Nein		
Single Level of Abstraction	☐Ja ☐Nein		
Interface Segregation Principle	☐Ja ☐Nein		

```java
public class PhiitFeed {
   private List<BlogEntry> entries;

   public PhiitFeed(User user) {
      for (int i = 0; i < entries.size(); i++) {
         BlogEntry entry = entries.get(i);
         if (entry.getMedia() != null) {
            if (user.getCredit() >= entry.getMedia().requiredTokens) {
               System.out.println(entry.getMedia().show(user));
            }
         }
         if (entry.getChat() != null) {
            LiveChat chat = entry.getChat();
            for (int j = 0; j < chat.messages.size(); j++) {
               System.out.println(chat.getMessages());
            }
         }
      }
   }
}
```

```java
public class User {
   private int credit;
   public void reloadCredit(int increasement) { credit += increasement; }

   public int getCredit() { return this.credit; }
}
```

```java
public abstract class BlogEntry {
   private Media media;
   private LiveChat chat;

   public Media getMedia() { return media; }

   public LiveChat getChat() { return chat; }
}
```

```java
// [George, 4-1-2001]: wtf - please stop programming
public abstract class Media {
   public final int requiredTokens = 2;

   // TODO: Implement this method in its subclasses
   public List<Image> showUp() { return List.of(); }

   public List<Image> show(User user) {
      if (user.getCredit() > requiredTokens) {
         user.reloadCredit(-requiredTokens);
         return this.showUp();
      }
      Payment.suggestRefill(requiredTokens - user.getCredit());
      return null;
   }
}
```

```java
public class Image extends Media {
   private Image image;

   public List<Image> showUp() { return List.of(image); }
}
```

```java
public class Video extends Media {
   private final List<Image> frames = new ArrayList<>();

   public List<Image> showUp() { return frames; }
}
```

```java
public class LiveChat {
   public final List<String> messages = new ArrayList<>();

   void sendMessage(int credit, String message) {
      int costs = Math.round((float) message.length() / 100);
      if (credit < costs) {
         Payment.suggestRefill(costs - credit);
      } else { messages.add(message); }
   }

   public String getMessages() { return messages.get(messages.size() - 1); }
}
```

```java
public class Payment {
   public static void suggestRefill(int suggestedRefill) {
      System.out.println("You should buy " + suggestedRefill + " more tokens in our shop!");
   }
}
```

Aufgabe 6: Softwarequalität (8 Punkte)

Im weiteren Verlauf der Softwareentwicklung beschließt ihr Auftraggeber die Meldung von unangemessenen Inhalten komplett zu automatisieren. Sobald ein Inhalt als unangemessen klassifiziert oder von anderen Nutzern gemeldet wird, soll dieser sofort unzugänglich gemacht werden. Das Teilsystem soll als zuverlässiges Echtzeitsystem implementiert werden.

a) Handelt es sich bei dem Meldesystem um ein weiches oder hartes Echtzeitsystem (*soft/ hard real-time system*)? Erklären Sie die beiden Begriffe und begründen Sie kurz. (2 Punkte)

b) Ist jedes Echtzeitsystem ein zuverlässiges System? Begründen Sie Ihre Antwort kurz. (1 Punkt)

c) Im Meldesystem können an verschiedenen Stellen Fehler auftreten. Erklären Sie die verschiedenen Fehlerarten Fault, Error und Failure und ihren Zusammenhang. Machen Sie jeweils ein Beispiel für das Meldesystem. (5 Punkte)

Hinweis: Die Klassifizierung unangemessener Inhalte basiert entweder auf Nutzermeldungen oder auf einer Bildklassifikation. Sie können sich einen zweistufigen Prozess vorstellen, bei dem erst die dargestellten Elemente auf einem Bild klassifiziert werden. In einem zweiten Schritt werden die Kombinationen der klassifizierten Elemente bezüglich Unangemessenheit klassifiziert.