Erstellung des Product Backlogs/ Anforderungsliste: Die Anforderungen an das System sollten in Form eines Product Backlogs erfasst werden. Hierbei sollten sowohl funktionale als auch nicht-funktionale Anforderungen berücksichtigt werden.

Sprint Backlog: Planung von Sprints in regelmäßigen Abständen, bei denen bestimmte Funktionen des Online-Quizsystems entwickelt und umsetzt werden. Erstellenung eines Backlog für jeden Sprint ein eigenes Sprints, in dem die Anforderungen aus dem Product Backlog priorisiert und auf die Sprints aufgeteilt werden.

Risikoliste: Erstellung eines Risikoliste, in der alle potenziellen Risiken für das Projekt aufgeführt werden. Hierbei sollten insbesondere Risiken berücksichtigt werden, die sich auf die Entwicklung und Umsetzung des Online-Quizsystems auswirken könnten, wie z.B. mangelnde Ressourcen oder technische Schwierigkeiten.

Risiko-Wert-Matrix: Zuordnung eines Wertes jedes Risikos in der Risikoliste, der den Grad der Wahrscheinlichkeit und den Grad des möglichen Schadens angibt. Verwendung dieser Informationen, um die Risiken nach ihrer Priorität zu ordnen und Maßnahmen zu entwickeln, um die Risiken zu minimieren oder zu vermeiden.

Testprotokoll: Erstellung eines Testprotokoll, in dem alle Tests dokumentiert werden, die während der Entwicklung des Online-Quizsystems durchgeführt wurden. Hierbei sollten alle Tests, Ergebnisse und Feedbacks der Nutzer aufgeführt werden, um sicherzustellen, dass das System korrekt funktioniert und den Anforderungen der Nutzer entspricht.

Ergebnisdokumentation: Dokumentierung der Ergebnisse und Fortschritte, die während der Entwicklung des Online-Quizsystems erzielt wurden. Hierbei sollten auch Änderungen am Konzept und am System selbst aufgeführt werden, um sicherzustellen, dass alle Teammitglieder und Stakeholder über den Fortschritt des Projekts informiert sind.

Benutzerhandbuch: Erstellen Sie ein Benutzerhandbuch, in dem die Bedienung und Funktionsweise des Online-Quizsystems beschrieben wird. Hierbei sollten alle Funktionen und Einstellungen des Systems verständlich und präzise beschrieben werden, um den Nutzern einen einfachen Einstieg in die Bedienung des Systems zu ermöglichen. Es empfiehlt sich, das Benutzerhandbuch regelmäßig zu aktualisieren, um sicherzustellen, dass es immer auf dem neuesten Stand ist.

Vorgehensmodell: Entscheiden Sie sich für ein geeignetes Vorgehensmodell für Ihr Projekt, z.B. Wasserfallmodell oder Agile Methoden wie Scrum oder Kanban. Legen Sie fest, welche Phasen und Schritte das Modell umfasst und wie diese durchgeführt werden sollen. Hierbei sollten Sie sicherstellen, dass das Vorgehensmodell den Anforderungen des Projekts entspricht und dass es für alle Teammitglieder verständlich und umsetzbar ist.

Rollenverteilung: Definieren der Rollen und deren Verantwortlichkeiten aller Teammitglieder im Projekt. Festlegung der Übernahme der Aufgaben und für die Umsetzung der verschiedenen Phasen und Schritte des Vorgehensmodells. Sicherstellung der Rollen und der klar definierten Verantwortlichkeiten. Darstellung in einer Tabelle möglich.

Architektur: Beschreibung der Systemarchitektur, Komponenten und deren Zusammenhänge.

Identifikation von Komponenten und deren Funktionen

Erstellung von Diagrammen zur Veranschaulichung der Architektur (z.B. UML-Diagramme)

Dokumentation von Schnittstellen und Datenflüssen zwischen den Komponenten

Prozessbeschreibung festlegen:

Eine Prozessbeschreibung wird festgelegt, um den Ablauf der Entwicklung und die Beteiligung der verschiedenen Akteure zu beschreiben.

Erstellung von EPKs (Ereignisgesteuerte Prozessketten), um die Prozesse zu modellieren

Verwendung von UML (Unified Modeling Language) zur Modellierung von Prozessen

Beschreibung von Abläufen und Workflows im System

Modelle erstellen: Je nach Art der Software können verschiedene Modelle zur Beschreibung der Funktionalität und der Datenstruktur verwendet werden.

Erstellung von ERM-Diagrammen (Entity-Relationship-Model), um die Datenmodelle zu modellieren

Dokumentation von Attributen und Beziehungen zwischen den Entitäten

Beschreibung von Datenflüssen und Datenverarbeitung

Verwendete Programmiersprachen und Modelle festlegen: Die verwendeten Programmiersprachen und Modelle werden festgelegt, um einen Überblick über die technischen Details des Projekts zu geben. Hier können z.B. Programmiersprachen wie Java oder Python, Frameworks wie React oder Vue.js und Datenbanken wie MySQL oder MongoDB genannt werden.

Auflistung der verwendeten Programmiersprachen und Frameworks

Beschreibung von Schnittstellen und API (Application Programming Interface)

Dokumentation von Konfigurationsdateien und Einstellungen

Eingesetzte Werkzeuge beschreiben: Eine Beschreibung der eingesetzten Werkzeuge wie Entwicklungsumgebungen, Versionskontrollsysteme, Continuous Integration/Deployment.

Auflistung der verwendeten Entwicklungsumgebungen (IDEs)

Dokumentation von Versionskontrollsystemen (VCS)

Beschreibung von Build-Tools und Deployment-Prozessen

Qualitätsziele: Hier sollten die Ziele definiert werden, die das System erreichen soll, um als qualitativ hochwertig zu gelten. Beispiele könnten sein: Benutzerfreundlichkeit, Skalierbarkeit, Performance, Stabilität, Sicherheit, etc.

Qualitätsplanung: Auf Basis der definierten Qualitätsziele sollte ein Plan erstellt werden, wie diese Ziele erreicht werden sollen. Dabei sollten auch relevante Normen und Standards berücksichtigt werden, die für das System gelten.

Konstruktives Qualitätsmanagement: Hierbei geht es darum, während des Entwicklungsprozesses sicherzustellen, dass die Qualitätsziele und -pläne umgesetzt werden. Dazu können beispielsweise Code-Reviews, Pair-Programming, Coding-Guidelines oder automatisierte Tests eingesetzt werden.

IT-Sicherheit: Ein wichtiger Aspekt bei der Entwicklung von Software ist die IT-Sicherheit. Hier sollten die relevanten Anforderungen an die Sicherheit definiert werden und entsprechende Maßnahmen ergriffen werden, um diese Anforderungen zu erfüllen. Dazu können beispielsweise Verschlüsselung, Zugriffskontrollen oder Penetrationstests gehören.

Statische Qualitätssicherung: Hierbei handelt es sich um Maßnahmen, die während der Entwicklungsphase durchgeführt werden, um mögliche Fehler im Code zu identifizieren. Beispiele für statische Qualitätssicherung sind Code-Reviews, automatisierte Code-Analysen oder Coding-Guidelines.

Review-Verfahren: Reviews sind ein wichtiger Bestandteil der Qualitätssicherung. Hierbei werden beispielsweise Anforderungen, Architektur-Entwürfe oder Code von anderen Teammitgliedern überprüft und Feedback gegeben.

Testfallerstellung: Die Erstellung von Testfällen ist ein wichtiger Schritt, um sicherzustellen, dass das System fehlerfrei funktioniert. Hierbei sollten alle relevanten Funktionalitäten des Systems abgedeckt werden. Es können beispielsweise automatisierte Tests oder manuelle Tests durchgeführt werden.