Hochschule für Technik und Wirtschaft Berlin

University of Applied Sciences

Pflichtenheft und technische Spezifikation im Programmierprojekt

<<WeGo Gym>>

Mitarbeiter: <<Ahmad Amairy, Jef Arthur Nanfack Tonfack, Zia Ul-Mostafa, Mohamad Rashad Mardini>>

<<Projektlogo>>

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

Inhaltsverzeichnis

Hochschule für Technik und Wirtschaft Berlin
University of Applied Sciences

1 Visionen und Ziele

Beschreiben Sie die Visionen und Ziele für das Programmierprojekt. Stellen Sie dabei die wesentlichen Dinge des Lastenheftes dar und beschreiben Sie aus Anbietersicht den Nutzen, der mit dem zu realisierenden System erzielt werden soll.

Hier zeigt der Anbieter, dass er das Problem des Auftraggebers verstanden hat und die einzelnen Punkte auch in der richtigen Priorisierung erfasst hat.

Unsere Vision für das Gymmanagement-Programm ist es, ein umfassendes, benutzerfreundliches und effizientes System zu entwickeln, das die Geschäftsprozesse des Fitnessstudios optimiert und eine positive Kundenerfahrung bietet.

Das Programm deckt alle Aspekte des Fitnessstudios ab, einschließlich Mitgliederverwaltung,Kurs-und Terminplanung, Personalmanagement. Die intuitive Oberfläche erleichtert die Umgang für Mitarbeiter und Kunden.

Das System bietet eine effiziente Verwaltung von Mitgliederdaten, Abonnements und Vertragsbedingungen sowie eine intelligente Planung und Verwaltung von Kursen und Trainingszeiten.

2 Anforderungsanalyse

2.1 Funktionale Anforderungen / Use-Cases

Beschreiben Sie die wesentlichen funktionalen Anforderungen bzw. Use-Cases Ihres Systems. Dies können Sie als Use-Case-Diagramm (geeignet bei mehreren Typen von Anwendern der Software) oder als Tabelle mit funktionalen Anforderungen darstellen.

Legen Sie fest, welches Teammitglied für welchen Use-Case bzw. welche funktionale Anforderung verantwortlich ist. In diesem Abschnitt dreht sich alles um die Anforderungen aus Nutzersicht. Eine Definition von internen Abläufen ist hier nicht sinnvoll.

Alle funktionalen Anforderungen bzw. Use-Cases sollten nummeriert werden, damit man sich im Abschnitt 4 darauf beziehen kann.

Verantwortliches Teammitglied: Ahmad Amairy

Realisierte Use-Cases : Kurs und event leiten(Verwaltung), Pläne erstellen und leiten(Verwaltung)

Verantwortliches Teammitglied: Zia Ul-Mostafa

Realisierte Use-Cases : Profile verwalten(Verwaltung), Support leisten(Verwaltung)

Verantwortliches Teammitglied: Mohamad Rashad Mardini

Realisierte Use-Cases : Profile bearbeiten(Kunden), Support fragen(Kunden)

Verantwortliches Teammitglied: Jef Arthur Nanfack Tonfack

Realisierte Use-Cases : Kurs und Event teilnehmen(Kunden), Pläne erstellen(Kunden)

2.2 Nicht-funktionale Anforderungen

Führen Sie an dieser Stelle alle nicht-funktionalen Anforderungen Ihres Systems auf. Zum Beispiel unter welchem Betriebssystem die Software laufen soll, oder welche Anforderungen an Antwort- oder Verarbeitungszeiten gestellt werden.

Die nicht-funktionalen Anforderungen des Gymmanagement-Programms umfassen folgende Aspekte:

Betriebssystem-Kompatibilität: Die Software soll auf Windows, macOS und gängigen Linux-Distributionen lauffähig sein.

Performance: Die Antwortzeit sollte unter einer Sekunde und die Verarbeitungszeit unter zwei Sekunden liegen.

Skalierbarkeit: Das System soll mit wachsenden Mitgliederzahlen und Geschäftsanforderungen skalieren.

Zuverlässigkeit: Die Systemverfügbarkeit sollte mindestens 99,9 % betragen, um Ausfälle zu minimieren.

Sicherheit: Strenge Sicherheitsstandards wie Verschlüsselung und Zugriffskontrollen sind erforderlich.

Benutzerfreundlichkeit: Intuitive Bedienung mit klarer Navigation und verständlichen Anweisungen.

Wartungsfähigkeit: Leicht zu warten und zu aktualisieren, um Aktualisierungen und Fehlerbehebungen zu erleichtern.

Interoperabilität: Nahtlose Integration mit anderen Systemen wie Buchungssystemen und Zahlungsgateways.

Dokumentation: Bereitstellung klarer Handbücher und technischer Details.

Internationalisierung: Unterstützung von Mehrsprachigkeit und Anpassung an lokale Standards.

Zugänglichkeit: Einhaltung von Standards der Barrierefreiheit, um auch Menschen mit Behinderungen die Nutzung zu ermöglichen.

2.3 Risiken

Beschreiben Sie mögliche Risiken für die erfolgreiche Umsetzung Ihres Projektes! Welche Maßnahmen können Sie ergreifen, um die Risiken zu reduzieren? Diese Darstellung sollte in einer Tabelle erfolgen.

Das Gymmanagement-Programm birgt einige potenzielle Risiken, die die erfolgreiche Umsetzung des Projekts beeinflussen könnten. Um diese Risiken zu minimieren, ergreifen wir folgende Maßnahmen:

Verzögerungen im Zeitplan: Unvorhergesehene Herausforderungen oder Abhängigkeiten könnten zu Verzögerungen führen. Wir setzen auf eine realistische Zeitplanung, regelmäßige Überprüfungen und Anpassungen des Projektplans.

Mangelnde Qualität: Schlechte Softwarequalität kann zu Fehlfunktionen und unzufriedenen Kunden führen. Wir führen daher gründliche Tests, Qualitätskontrollen und Benutzerrückmeldungen durch.

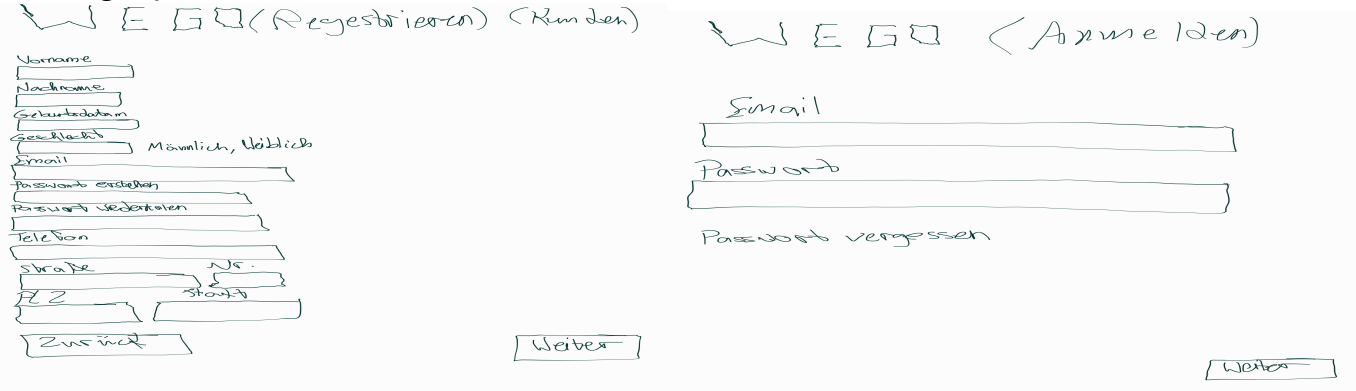
Sicherheitsrisiken: Datenverletzungen oder unautorisierter Zugriff können auftreten. Wir setzen auf robuste Sicherheitsmaßnahmen und Datenschutzrichtlinien, einschließlich Verschlüsselung und Zugangskontrollen.

Änderungen im Projektumfang: Änderungen im Anforderungsprofil könnten das Projekt beeinträchtigen. Wir definieren klare Anforderungen, führen Change-Management-Prozesse durch und kommunizieren regelmäßig mit den Stakeholdern.

Technologische Herausforderungen: Schwierigkeiten bei der Integration oder Nutzung neuer Technologien begegnen wir mit sorgfältiger Planung, Prüfung und Zusammenarbeit mit erfahrenen Partnern.

2.4 GUI

Erstellen Sie einen Mockup Ihrer GUI. Dazu sollen für die wichtigen Anwendungsfälle die Oberflächen entworfen und ihre Funktion beschrieben werden. Das Mockup kann mit einem beliebigen Tool umgesetzt werden (Powerpoint, inkscape, Figma, Paint, WPF, egal!).

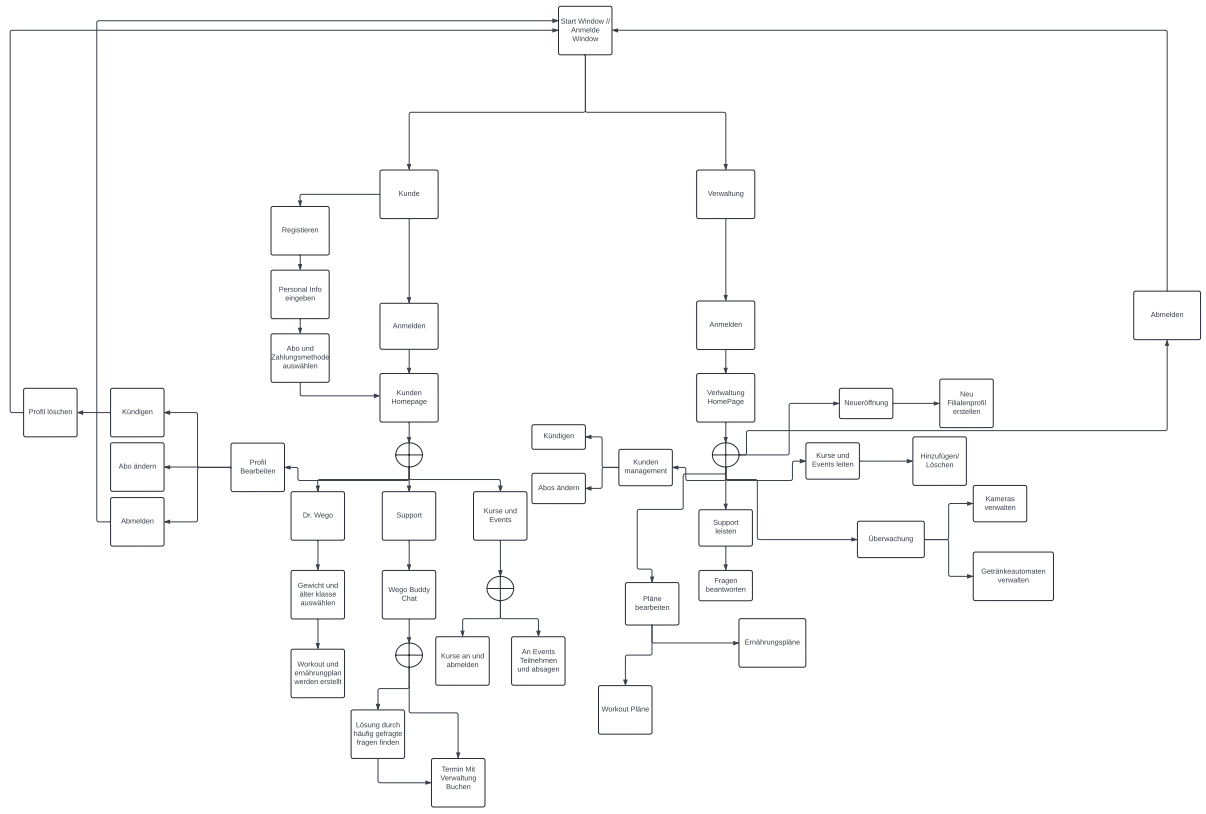


Realisierung

3.1 Komponenten

Legen Sie die für Ihr System zu erstellenden Komponenten fest. Ordnen Sie die Komponente dem verantwortlichen Teammitglied zu. Legen Sie fest, welche Use-Cases in welcher Komponente oder welchen Komponenten aus heutiger Sicht zu realisieren sind.

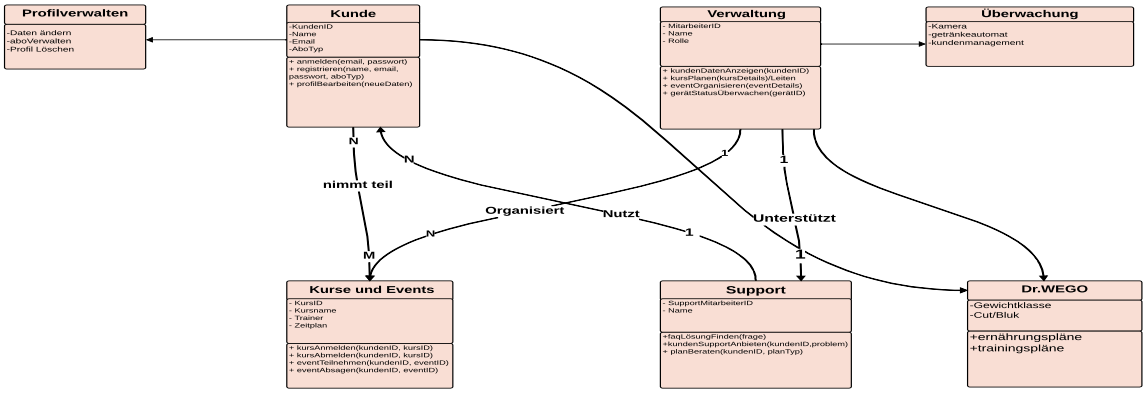
Erstellen Sie ggfs. ein Komponenten-Diagramm des zu realisierenden Programms mit den dazu erforderlichen Komponenten und zeigen Sie die Abhängigkeiten zwischen den Komponenten auf.



3.2 Interne Schnittstellen / Klassendiagramm

Erstellen Sie ein Klassendiagramm, um die zu realisierenden Schnittstellen konkret zu spezifizieren. Überlegen Sie, welche Funktionen von welcher Klasse bereitgestellt werden müssen und ***achten Sie auf eine sinnvolle Softwarearchitektur und ggfs. passende Entwurfsmuster (siehe Programmierung 2)****.*

Singleton für die Datenbankverbindung oder Factory für die Instanzerstellung könnten ebenfalls hilfreich sein, außerdem ermöglicht uns Singleton einen globalen Zugriffsmöglichkeiten.



4 Entwicklungs- und Teststrategie

Für eine unabhängige Entwicklung in Phase 2 benötigen Sie eine geeignete Entwicklungs- und Teststrategie, welche in diesem Abschnitt dargelegt werden muss.

In diesem Abschnitt können Sie daher aufführen, ob Sie Unittests verwenden oder welche anderen Arten von Funktionalitätstests Sie durchführen. Sie sollten zu jedem Use-Case geeignete Tests definieren, um die funktionalen Anforderungen zu überprüfen.

Weiterhin können Sie hier erörtern, ob Sie für die Entwicklung auf mehrere Branches zurückgreifen wollen.

Für die unabhängige Entwicklung in Phase 2 des Gymmanagement-Programms verwenden wir eine Entwicklungs- und Teststrategie, die die Qualität und Zuverlässigkeit der Software sicherstellt.

Entwicklungsstrategie

Versionierung und Branching: Nutzung von Git mit verschiedenen Branches (z. B. Hauptzweig, feature branches, bugfix branches) für parallele Entwicklungsarbeiten und Code-Überprüfungen.

Kontinuierliche Integration (CI): Automatisiertes Testen und Überprüfen bei jedem Push in einen Entwicklungszweig.

Kontinuierliche Bereitstellung (CD): Nach erfolgreichen Tests und Überprüfungen können Änderungen in den Produktionszweig bereitgestellt werden.

Teststrategie

Unittests: Testen einzelner Code-Einheiten für jeden Use-Case.

Integrationstests: Prüfen des Zusammenspiels verschiedener Komponenten und Use-Cases.

Systemtests: Überprüfen des gesamten Systems, einschließlich funktionaler und nicht-funktionaler Anforderungen.

Akzeptanztests: Durchführung mit Auftraggeber und Endbenutzern, um Anforderungen zu erfüllen.

Regressionstests: Sicherstellung, dass keine vorhandene Funktionalität beeinträchtigt wurde.

End-to-End-Tests: Simulation typischer Benutzerszenarien, um die Funktionalität des gesamten Systems zu überprüfen.

Durch diese Tests stellen wir sicher, dass alle funktionalen Anforderungen der Use-Cases erfüllt werden. Die Entwicklungsstrategie mit mehreren Branches ermöglicht eine effiziente und unabhängige Entwicklung in Phase 2 des Projekts.

5 Kooperationen und Verwertungsplan

Arbeiten Sie mit anderen Teams aus anderen Studiengängen zusammen? Welchen Fokus hat Ihre Kooperation? Welche Aspekte dieser Kooperation konnten Sie für Ihr Konzept bisher verwenden, welche planen Sie zu verwenden? Gibt es einen regelmäßigen Austausch?

Gibt es einen Verwertungsplan für Ihr Softwareprodukt? In diesem Abschnitt können Sie kreativ sein und nicht-technologische Aspekte Ihrer Software betrachten.

Legen Sie im Team fest, ob ihre erstellte Software anschließend „open source" ist bzw. von wem sie nachgenutzt werden kann. Eine übliche Lizenz für Hochschulprojekte ist die MIT-Lizenz.

Kooperationen:

Design: Zusammenarbeit mit Design-Studierenden für eine benutzerfreundliche und attraktive Benutzeroberfläche.

Wirtschaft: Austausch mit Wirtschaftsstudierenden, um Markttrends zu berücksichtigen und erfolgreiche Vermarktungsstrategien zu entwickeln.

Kommunikation: Zusammenarbeit mit Kommunikationswissenschaftlern für effiziente Kundeninteraktionen.

Recht: Beratung durch Jurastudierende, um Datenschutz- und Rechtsanforderungen zu erfüllen.

Aspekte der Kooperation:

Benutzererfahrung: Design-Input führte zu einer intuitiven Oberfläche.

Marktanalyse: Wirtschaftliche Perspektiven halfen bei der Integration von Marktanforderungen.

Kommunikationsstrategie: Unterstützung bei der Entwicklung effektiver Kundenkommunikationskanäle.

Rechtliche Aspekte: Juristische Beratung gewährleistete Compliance.

Verwertungsplan:

Austausch: Regelmäßige Treffen und Feedback-Runden mit anderen Teams.

Verwertung: Langfristige Lizenzverträge mit Fitnessstudios geplant.

Open Source: Veröffentlichung des Projekts unter der MIT-Lizenz, um breite Nachnutzung und Weiterentwicklung zu fördern. Unsere Zusammenarbeit mit anderen Disziplinen hat das Konzept und die Qualität des Gymmanagement-Programms verbessert. Die MIT-Lizenz ermöglicht eine offene Nutzung und Weiterentwicklung des Projekts durch andere.

EA-Diagramm