TAU INF202 Software Engineering

Pflichtenheft

Projektdokumentation

Version: 2023.04.16

Status: Entwurf

Messaging Applikation

Verantwortliche/r:
Eren Naci Odabaşı, e180501038@stud.tau.edu.tr
Nihat Akın, e180501008@stud.tau.edu.tr

Stakeholder: DI. Ömer Karacan, omer.karacan@tau.edu.tr

Dokumentenverwaltung

Dokument-Historie

Version	Status *)	Datum	Verantwortlicher	Änderungsgrund
v1.0	Entwurf	16.04.2023	Ö. Karacan	Vorlage wurde für die Studentenprojekte freigegeben

Dokument wurde mit folgenden Tools erstellt:
Microsoft Office Word

Inhaltsverzeichnis

Einleitung 4
Ausgangssituation und Ziele 5
Gesamtarchitektur 7
Funktionale Anforderungen 9
Nichtfunktionale Anforderungen 11
Abnahmekriterien 12
Projekt Meilensteine 13
Referenzen 14

Einleitung

Echtzeit-Messaging-Applikationen wie Whatsapp und Telegram erleichtern unser tägliches Leben im Hinblick auf eine schnelle Kommunikation. Professionellere Anwendungen wie Microsoft Teams werden jedoch in der Regel in Unternehmen eingesetzt.

Um eine Messaging-Anwendung zu entwickeln, die den oben genannten ähnlich ist, benötigt man:

1. eine Datenbank, in der die Informationen der Benutzer gespeichert werden,
2. eine einfache und nützliche Schnittstelle, über die die Benutzer ihre Nachrichten senden können,
3. einen Server, der Gruppenchats und Peer-to-Peer-Chat-Funktionen bereitstellen kann.

Dieses Dokument beschreibt die Motivation für die Entwicklung einer Echtzeit-Messaging-Anwendung, ihre Funktionen, die Zielgruppe, Systemumfeld, Rahmenbedingungen, Ziele und Lösungen, Architektur von dem System, Use Cases, Technischen und fachlichen Anforderungen, Datenmodell, und Nichtfunktionale Anforderungen.

Ausgangssituation und Ziele

Mit der Zunahme der Fernarbeit und des Fernunterrichts ist die Nachfrage nach Kommunikationstools gestiegen, die es Einzelpersonen und Teams ermöglichen, in Verbindung zu bleiben und in Echtzeit zusammenzuarbeiten. Es besteht daher ein Bedarf an einer Messaging-Anwendung, die schnell, zuverlässig, funktionsreich und sicher ist.

Eine Echtzeit-Messaging-Anwendung kann eine zuverlässige, funktionsreiche und sichere Plattform für die effektive Kommunikation und Zusammenarbeit von Einzelpersonen und Teams bieten. Darüber hinaus kann eine gut konzipierte Messaging-Anwendung einen Wettbewerbsvorteil auf dem überfüllten Markt der Kommunikationstools bieten. Daher ist die Entwicklung einer Echtzeit-Messaging-Anwendung ein vielversprechendes Projekt mit dem Potenzial, ein dringendes Problem zu lösen und den Benutzern einen wertvollen Dienst zu bieten.

Einleitung

In dieser Kapital wurden die Problemstellung , die Zielgruppe (Stakeholder), Systemumfeld, Rahmenbedingungen, Qualitative und quantitavie Ziele dieser Messaging-App beschrieben.

Problemstellung (Funktionalität)

Messaging-Anwendungen, die gezwungen sind, das globale Netzwerk zu nutzen, haben Probleme wie langsame Nachrichtenübermittlung, Sicherheitsprobleme, Ineffiziente Kommunikation. Diese Messaging-Anwendung, die ein lokales Netz nutzt, bietet eine sichere Messaging-Umgebung, die besser gegen externe Hackerangriffe geschützt ist und ihre Funktion auch dann aufrechterhalten kann, wenn das globale Netz unterbrochen ist.

Stakeholder (Anwender):

Die Hauptzielgruppe dieses Projekts sind Unternehmen, die Informationen in einer Gruppe oder individuell durch Nachrichtenübermittlung innerhalb des Unternehmens austauschen müssen. Allerdings ist es vielleicht nicht ganz richtig, den Anwendungsbereich des Projekts auf Unternehmen zu beschränken. Denn wir alle nutzen im täglichen Leben Echtzeitanwendungen mit Gruppen- und Einzelchatfunktionen. Mit anderen Worten: Einzelpersonen, Teams in jeder Organisation und viele ähnliche Szenarien, die lokales Kommunikation nutzen können, können zur Zielgruppe dieses Projekts gehören.

Systemumfeld (Einsatzumgebung)

Die Anwendung ermöglicht die Nutzung von Gruppen- und Peer-to-Peer-Chat-Funktionen durch die Verwendung von verschiedenen Computern, die mit demselben Internet-Netzwerk verbunden sind, d.h. mit Zugang zu einem lokalen Server.

Rahmenbedingung (Einschränkungen)

Die Echtzeit-Messaging-Anwendung sollte unter dem Aspekt der Skalierbarkeit konzipiert und entwickelt werden, da sie große Mengen an Datenverkehr und Benutzern bewältigen muss. Deswegen ist vor dem Einsatz der Echtzeit-Messaging-Anwendung es wichtig, gründliche Simulationen und Tests durchzuführen, um mögliche Fehler, Leistungsprobleme oder Sicherheitslücken zu ermitteln. Die Tests sollten verschiedene Szenarien und Nutzungsmuster umfassen, um sicherzustellen, dass die Anwendung unterschiedliche Verkehrsaufkommen und Benutzeraktivitäten bewältigen kann.

Ein Proof-of-Concept-Prototyp kann entwickelt werden, um die Durchführbarkeit des Projekts zu testen und die Kernfunktionen der Messaging-Anwendung zu demonstrieren. Hier die Einsatzumgebung, wobei Faktoren wie Server-Hardware, Softwareanforderungen und Sicherheitsmaßnahmen zu berücksichtigen sind. Dann schließlich sollte die Anwendung die Benutzerdaten durch sichere Anmelde- und Authentifizierungsmechanismen implementieren und über geeignete Zugriffskontrollen verfügen, um unbefugten Zugriff oder Datenverletzungen zu verhindern.

Ziele (Lösung)

Qualitative und quantitative Ziele, die durch die Entwicklung dieser Echtzeit-Nachrichtenanwendung erreicht werden könnten:

Qualitative Ziele:

- Verbesserte Benutzerfreundlichkeit: Durch gute Konzipierung, dieser Messaging-App kann eine intuitive und nahtlose Benutzererfahrung bieten, die es den Benutzern leicht macht, in Echtzeit zu kommunizieren und zusammenzuarbeiten.
- Gesteigerte Zusammenarbeit und Produktivität: Einer der Ziele von dieser Projekt ist die Kommunikation und Zusammenarbeit zwischen Teammitgliedern erleichtern, was zu einer höheren Produktivität und Effizienz führt

Quantitative Ziele:

- Schnellere Nachrichtenübermittlung: Dieser Echtzeit-Messaging-App kann im Vergleich zu herkömmlichen Messaging-Anwendungen eine schnellere Zustellung von Nachrichten ermöglichen, wodurch Verzögerungen verringert und die Antwortzeiten verbessert werden.
- Erhöhte Benutzerbindung: Durch die Bereitstellung dieser funktionsreichen und zuverlässigen Messaging-App ist es möglich, das Engagement und die Bindung der Benutzer über einen längeren Zeitraum zu erhöhen.

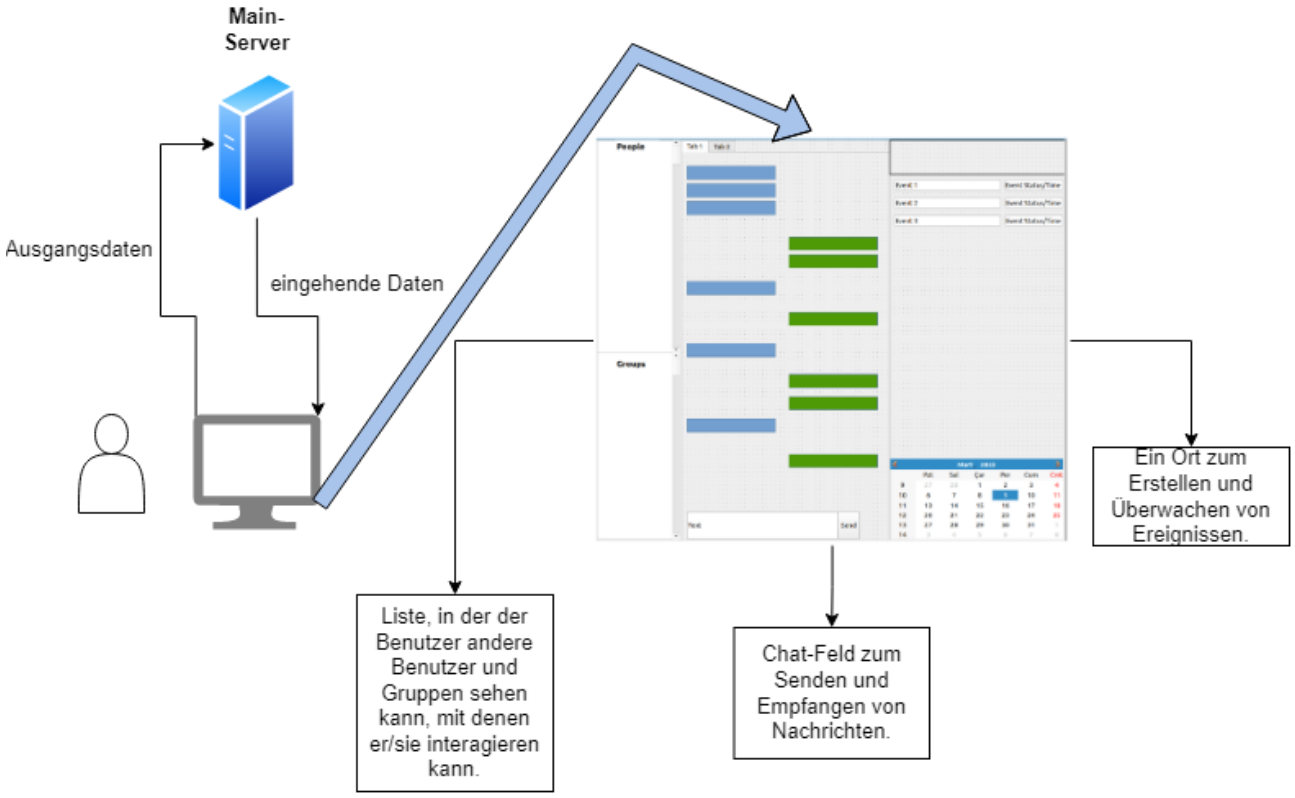
Gesamtarchitektur

Einleitung

Dieses Kapitel sind die Funktionalität des User-Systems und die Komponenten, die die Kommunikation zwischen verschiedenen Benutzern herstellt, definiert.

Gesamtarchitektur

Die Gesamtarchitektur besteht aus:



Komponente DA

Die Komponente DA (Desktop-Anwendung) stellt sowohl die QoL-Dienste für den Benutzer als auch das Kommunikationsmedium zwischen dem Hauptserver und der Anwendung dar.

Komponente MS

Die Komponente MS (Main-Server) verfügt über die notwendigen Algorithmen für die Kommunikation mit den Clients und die Weiterleitung der verschiedenen Anfragen.

Externe Schnittstellen

Keine externe Schnitttelle ist benötigt.

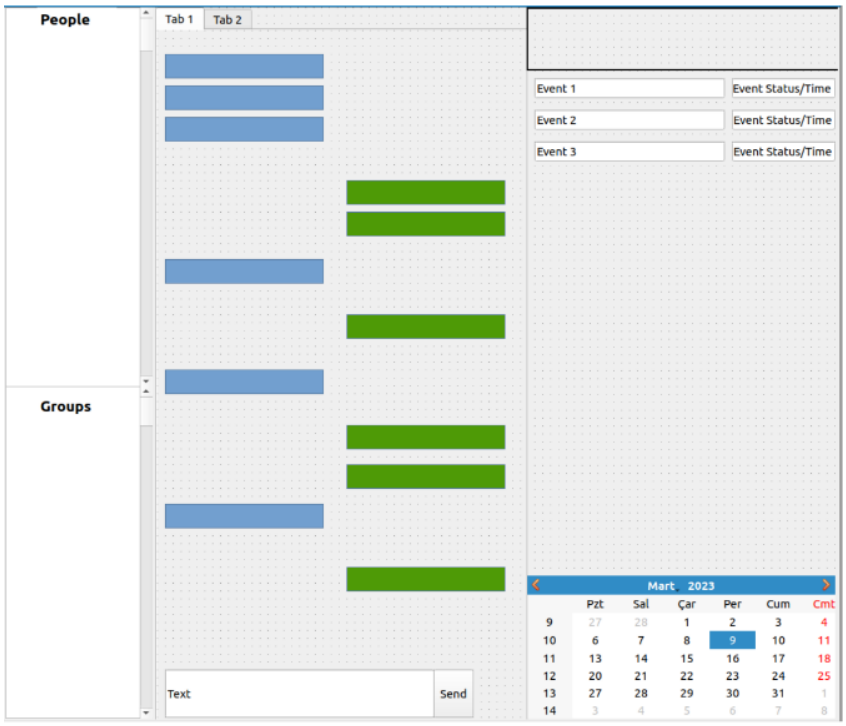
Funktionale Anforderungen

Einleitung

In diesem Kapitel sind die Anforderungen des Systems, z.B die Use Cases, definiert.

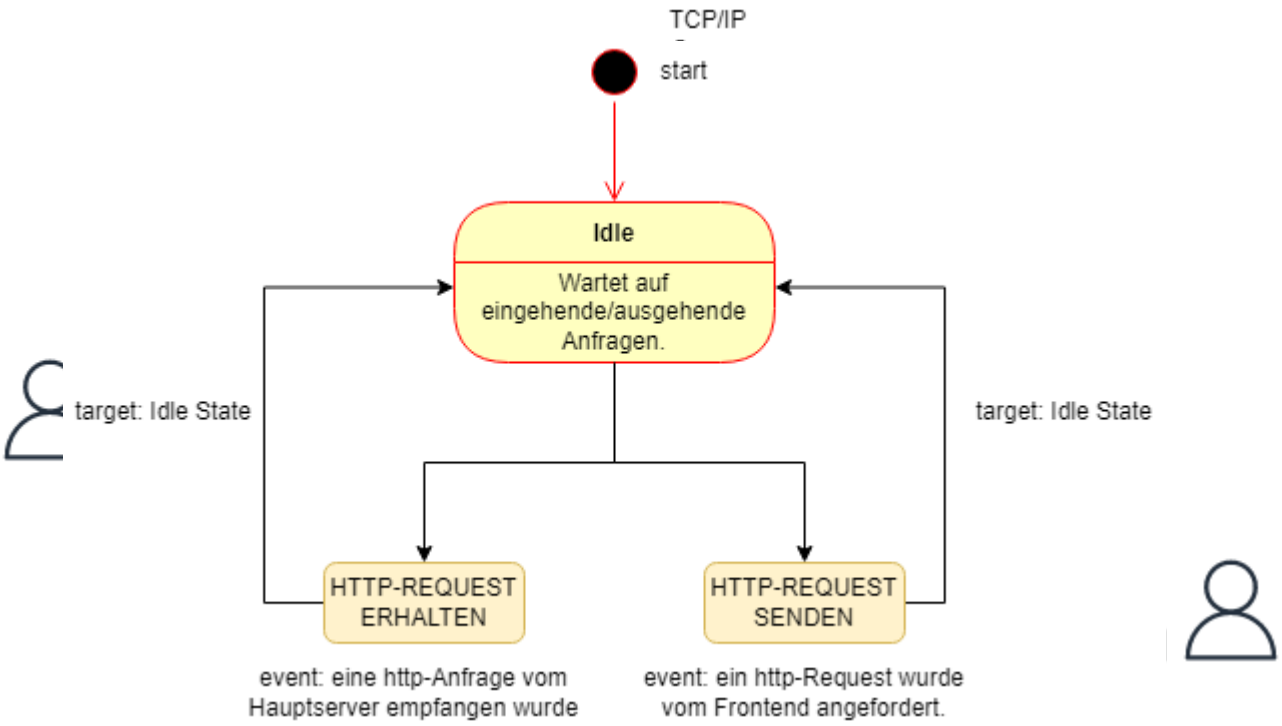
Use Cases

- Jede Unterfunktionalität (Chats, Kontakte, Ereignisse ...) sollte ihren eigenen Platz in der grafischen Oberfläche haben.
- Die Benutzeroberfläche soll konzeptionell nach dem Prototyp entwickelt werden (Abb.).
- Im Bereich Kontakte müssen alle Kontakte des Benutzers sowie die registrierten Gruppen angezeigt werden. Ein neuer oder ein bestehender Chat sollten sich im Chat-Bereich öffnen, wenn mit einem Kontakt interagiert wird.
- Im Chat-Bereich sollen alle geöffneten Chats angezeigt sein. Zwischen diese Chats sollten man wechseln können.
- Es sollte möglich sein, neue Kontakte über Einladungen hinzuzufügen und Gruppen mit mehr als zwei Benutzern zu erstellen.



Technischen und fachliche Anforderungen

- Das Frontend-System sollte als Client für den Hauptserver fungieren, wo es HTTP-Anfragen über das TCP/IP-Protokoll sendet und empfängt.
- Das Backend (Hauptserver) sollte in der Lage sein, alle von den Clients kommenden Anfragen asynchron zu bearbeiten.
- Alle Informationsanfragen sollten über den Hauptserver und die Datenbank abgewickelt werden. Das Frontend sollte keinen vollständigen Zugriff auf die DB haben.



Datenmodel

- Die Daten eines jeden Benutzers, der auf dem Hauptserver registriert ist, sollten in einer externen Datenbank gespeichert werden.
- Daten zu den Chats (Chatverlauf) können in einer lokalen SQL-Datenbank gespeichert werden, die von der Desktop-Anwendung verwaltet wird.

Nichtfunktionale Anforderungen

Einleitung

In diesem Kapitel sind die nicht-funktionalen Anforderungen ans Gesamtsystem aber auch an die einzelnen Systemkomponenten definiert. Es wird besonders auf die Software Qualität Wert gelegt (Testing).

Nicht-funktionale Anforderungen an die Systemarchitektur (Architekturmuster, Deployment)

Nicht-funktionale Anforderungen an die Entwicklungsumgebung

- Die Entwicklungsumgebung ist ein UNIX-basiertes Betriebssystem.

Nicht-funktionale Anforderungen an die Entwicklungswerkzeuge (Sprache, IDE, Frameworks)

- Die Backend Applikation sollten mit Rust von Grund auf entwickelt werden.
- Die Frontend Applikation soll mit dem QT-Framework (C++) realisiert werden.
- Die daten sollen in einem SQL-Datenbank abgespeichert werden.

Nicht-funktionale Anforderungen an die Teststrategie (Qualitätssicherung)

- Jede Funktionalität sollte einzeln getestet werden.
- Als Test-Framework wird Google-Test (für C++) und sein Rust-Wrapper verwendet.

Abnahmekriterien

Das Projekt wird mit den folgenden Artefakten abgegeben:

- Dokumentation:
  - Pflichtenheft: INF202-Messaging_App-Pflichtenheft-2023-v1.x.docx
- Software
  - Link zu Github Projekt: https://github.com/Gruppe1-Fulya/messaging_app
- Evidenz:
  - System/Software-Demo via Videoclip:

Projekt Meilensteine

Folgende Meilensteine sind verbindlich definiert:

- Meilenstein #1: Das Lastenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.
- Meilenstein #2: Das Pflichtenheft ist fertiggestellt und mit dem Stakeholder abgestimmt.
- Meilenstein #3: Implementation der Systemarchitekturs.
    - Es wird eine robuste GUI entwickelt.
    - Ein externes (nicht-blockierendes) Backend für die GUI wird implementiert.
    - Der Hauptserver ist implementiert.
- Meilenstein #4:
    - Die Test-Cases wurden durch eine Testumgebung durchgeführt.
- Meilenstein #5:
    - Das Projekt ist abgegeben.

Referenzen

(1) (2022, October 25). Difference between client-server and peer-to-peer network. BYJUS. Retrieved April 16, 2023, from https://byjus.com/gate/difference-between-client-server-and-peer-to-peer-network/

(2) Cisco. (2022, March 15). What is network programming? Cisco. Retrieved April 16, 2023, from https://www.cisco.com/c/en/us/solutions/enterprise-networks/what-is-network-programming.html