

**AS3 Doctor Forum**

**Documentation**

|  |  |
| --- | --- |
| **Authors** | **Group 07** |
| **Pham Duc Tung – C1707L3937** |
| **Date** | **03/26/2022** |
| **Instructor** | **Nguyen Minh Quan** |

**Table of Content**

[1. Problem Definition 3](#_Toc48501067)

[1.1. Problem Abstraction 3](#_Toc48501068)

[1.2. The Current System 3](#_Toc48501069)

[1.3. The Proposed System 4](#_Toc48501070)

[1.3.1. Boundaries of the System 4](#_Toc48501071)

[1.3.2. Hardware and Software Requirements 4](#_Toc48501072)

[2. Customer Requirements Specification 4](#_Toc48501073)

[2.1. Users of the System 4](#_Toc48501074)

[2.2. System functions 5](#_Toc48501075)

[3. System Designs 8](#_Toc48501076)

[3.1. Entity Relationship Diagram 8](#_Toc48501077)

[3.2. Database Design 8](#_Toc48501078)

[3.3. System functions design 10](#_Toc48501079)

[**3.3.1. Use-case diagrams & Sequence diagrams** 10](#_Toc48501080)

[**3.3.2. Class diagrams** 27](#_Toc48501081)

[4. Tasks sheet 32](#_Toc48501082)

# Problem Definition

## Problem Abstraction

Our client is a doctor who is one of the famous Cardiac specialists in the country. As he is one of the famous specialists in the country, he generally used to visit some places for operating some patients with some critical cardiac conditions. He also used to get invitations from various colleges in the country for attending and giving the seminars, also he runs a college, which is one of the famous medical colleges in the country.

He wants an online application such as a discussion forum for the doctors in the world, so that if anyone is having any queries (for the junior doctors, etc.) or to discuss some new cases in different fields. He wants this application to be developed as this can bridge the gap between the various doctors available in the world and can know each other through this portal. Also, in case of any emergencies or for some special cases one can go through the application and search for the various doctors based on the specialization and the location whichever is suitable so that so many people can be saved.

AS3 Doctor Forum is a forum platform built for Web Interface that suits this problem best, offers the key deliverables:

* Anyone can **Register** an **Account** to use *AS3 Doctor Forum*
* Anyone can see the **Number of Users** using *AS3 Doctor Forum*
* Registered Users can create **Posts** and manage their **Posts** on *AS3 Doctor Forum*
* Registered Users can choose their posts to be either **Published** or **Draft**, they can publish or delete later
* Registered Users can **save** any **Posts** they feel helpful to read it later
* Registered Users can give public **comments** on any **Posts** on *AS3 Doctor Forum*
* Registered Users can create a beautiful **Profile** on *AS3 Doctor Forum*
* Registered Users can decide his Profile to be **Public** to everyone, or **Private** to himself
* **Administrator** can manage **Users Profile** on *AS3 Doctor Forum*
* **Administrator** can manage **Posts** on *AS3 Doctor Forum*

## The Current System

***AS3 Doctor Forum*** was designed in Microservices Architectural Style. It supports both reusable API and asynchronous Event-driven communication. Backed by .NET GraphQL, where interface was formed with React.

Functions implemented:

* Stateless Tokenized Authentication & Authorization with Refresh Token
* Pub/Sub System for Login counts & Registration counts
* Posts Management
* Users Management
* Profile Management

Superiorities:

* Simple – a great place to make questions and help answering others
* Synchronized – use on any device
* Fast – database and network layers are fully optimized, taking advantage of GraphQL
* Secure – stateless JWT makes things impossible to be broken

## The Proposed System

EasyChat aims to offer a simple chat system, serving end-user without any training. However, the proposed system is desired to support Notification system (which is not presenting), and a more powerful Media processor (which is tightly weak by the time this document issued). Also, an expand for a wider range of devices is a good improvement.

## Boundaries of the System

EasyChat brings best experience for anyone who wants a fast web chat platform.

## Hardware and Software Requirements

* Modern stable version browsers

Table

Description automatically generated

* JavaScript enabled

# Customer Requirements Specification

## Users of the System

Users should be aware of installing required software specified in **section 1.3.2**.

EasyChat is ease to use, there is no restriction or knowledge requirement.

## System functions

Input/Output, data extraction and data preservation of functions listed in **section 1.2** are about to be described. Meanwhile, each function’s workflow will be illustrated in Sequence Diagrams of **section 3.3**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Input** | **Output** | **Data** |
| Register Account | string first name  string last name  string email  number password  number confirmPassword | The user who sends request will have an account that can access to the forum | from *accounts* |
| This feature allows user to create his/her own account. | | | |
| Login Account | string email  string password | The user who sends request will be logged in to the account | from *accounts* |
| This feature allow user to login his/her account | | | |
| Fetch Posts |  | The user that is sending request will be announced with an empty Event the user that is receiving request will be announced with FriendRequestWithAccountInformation - *section 3.3.2* | to, from *account\_friends* from *accounts* |
| This feature presents all the published posts on the forum | | | |
| Fetch Highest Rating Posts |  | ChannelWithLatestMessage - *section 3.3.2* | to, from *account\_friends* to, from *channels* to, from *channel\_participants* to, from *channel\_messages* from *accounts* |
| This feature allows current user to fetch all the published posts on the forum from highest to lowest rating. | | | |
| Fetch Hot Posts |  | UnfriendResponseModel - section 3.3.2 | from, to account\_friends |
| This feature allows current user to fetch all the highest rating posts within the current week | | | |
| Fetch New Posts |  | ChannelWithLatestMessage - *section 3.3.2* | to, from *channels* to, from *channel\_participants* to, from *channel\_messages* from *accounts* |
| This feature allows current user to fetch all the published posts on the forum from newest to oldest | | | |
| Fetch Channels |  | ChannelWithLatestMessage[] - *section 3.3.2* | from *channels* from *channel\_participants* from *channel\_messages* from *accounts* |
| This feature allows current user to fetch all channels that he is participating in. EasyChat will add the user in each SocketRoom generated with Channel. | | | |
| Fetch Full Channel | number id | ChannelWithMessages - *section 3.3.2* | from *channel\_messages* from *accounts* |
| This feature allows current user to fetch full information of a specific Channel. | | | |
| Leave Channel | number channelId | LeaveChannelResponseModel - *section 3.3.2* | to *channel\_participants* |
| This feature allows current user to kick him out of specific Channel. After the user left, a record in *channel\_participants* will be removed, but his messages in that channel persist. He will also be removed from SocketRoom if online. | | | |
| Update Channel | number channelId string channelName avatarFile | Channel - *section 3.3.2* | to *channels* |
| This feature allows any user inside the Channel to update Channel. Avatar will be uploaded to Amazon S3. | | | |
| Fetch Channel Participants | number channelId | ChannelParticipantWithInformation[] - *section 3.3.2* | from *channel\_participants* from *accounts* |
| This feature allows current user to fetch all participants of a Channel. | | | |
| Add People to Channel | number channelId number[] memberIds | ChannelWithLatestMessage - *section 3.3.2* | to, from channel\_participants from *channels* from *channel\_messages* from *accounts* |
| This feature allows any user inside the Channel to add his friends to Channel. EasyChat will send Event to all member's online SocketClient in *memberIds* and add them all to the SocketRoom. | | | |
| Remove People from Channel | number accountId number channelId | LeaveChannelResponseModel - *section 3.3.2* | to *channel\_participants* |
| This feature allows any user inside the Channel to remove anybody inside Channel. EasyChat will send Event to removed account's online SocketClient and also remove them from the SocketRoom. His messages persist, but a record in *channel\_participants* will be erased. | | | |
| Send Chat message | number channelId string content number replyTo attachmentFile | ChatMessageFromDatabase - *section 3.3.2* | to, from *channel\_messages* from *accounts* |
| This feature allows current user to send message to a specific Channel. After message was successfully saved, EasyChat announces an Event to the Channel. Attachments will be uploaded to AmazonS3. | | | |
| Delete Chat message | number id | ChannelMessage - *section 3.3.2* | to *channel\_message* |
| This feature allows current user to unsent his message. After message was successfully saved, EasyChat announces an Event to the Channel. | | | |
| Update Profile | avatarFile | Account - *section 3.3.2* | to *accounts* |
| This feature allows current user to update his profile. There is only avatar update for now. Avatar file will be uploaded to Amazon S3. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Input** | **Output** | **Data** |
| Fetch Friends | number channelId | AccountFriendWithInformation - *section 3.3.2* | from *accounts* |
| This feature presents all ACCEPTED friends of the current logged in user. If *channelId* was provided, query friends that are outside of that Channel, otherwise, query all friends. | | | |
| Fetch Pending Friends |  | FriendRequestWithAccountInformation - *section 3.3.2* | from *accounts* from *account\_friends* |
| This feature presents all PENDING friends of the current logged in user with additional information. Such data can be explored in the Class Diagrams section. | | | |
| Add Friend | number friendId string message | The user that is sending request will be announced with an empty Event the user that is receiving request will be announced with FriendRequestWithAccountInformation - *section 3.3.2* | to, from *account\_friends* from *accounts* |
| This feature allows current user to create friend request to user specified by *friendId*, a greeting *message* is a must. | | | |
| Confirm Friend | number thePersonWhoSentRequestId | ChannelWithLatestMessage - *section 3.3.2* | to, from *account\_friends* to, from *channels* to, from *channel\_participants* to, from *channel\_messages* from *accounts* |
| This feature allows current user to accept a friend request. After confirming, EasyChat will change the flag in *account\_friends* and create new channel for the conversation. The new channel take the name of both two user's names. This channel will also be announced to both users with Event. The full operation is managed by transaction. | | | |
| Unfriend | number friendId | UnfriendResponseModel - section 3.3.2 | from, to account\_friends |
| This feature allows current user to remove a friend. This will also results in a friend deletion of the target. Channel among these guys stays, all messages persist, but they will no longer be able to add each other to any Channel. | | | |
| Create Channel | string name number[] participantIds string greetingMessage | ChannelWithLatestMessage - *section 3.3.2* | to, from *channels* to, from *channel\_participants* to, from *channel\_messages* from *accounts* |
| This feature allows current user to create records that forms in a Channel. Argument *participantIds* is optional, empty array means the group contains only him. The operation will be managed by transaction. After Channel created, EasyChat will send Event to all online SocketClient that are supposed to be inside and add them all to the SocketRoom. | | | |
| Fetch Channels |  | ChannelWithLatestMessage[] - *section 3.3.2* | from *channels* from *channel\_participants* from *channel\_messages* from *accounts* |
| This feature allows current user to fetch all channels that he is participating in. EasyChat will add the user in each SocketRoom generated with Channel. | | | |
| Fetch Full Channel | number id | ChannelWithMessages - *section 3.3.2* | from *channel\_messages* from *accounts* |
| This feature allows current user to fetch full information of a specific Channel. | | | |
| Leave Channel | number channelId | LeaveChannelResponseModel - *section 3.3.2* | to *channel\_participants* |
| This feature allows current user to kick him out of specific Channel. After the user left, a record in *channel\_participants* will be removed, but his messages in that channel persist. He will also be removed from SocketRoom if online. | | | |
| Update Channel | number channelId string channelName avatarFile | Channel - *section 3.3.2* | to *channels* |
| This feature allows any user inside the Channel to update Channel. Avatar will be uploaded to Amazon S3. | | | |
| Fetch Channel Participants | number channelId | ChannelParticipantWithInformation[] - *section 3.3.2* | from *channel\_participants* from *accounts* |
| This feature allows current user to fetch all participants of a Channel. | | | |
| Add People to Channel | number channelId number[] memberIds | ChannelWithLatestMessage - *section 3.3.2* | to, from channel\_participants from *channels* from *channel\_messages* from *accounts* |
| This feature allows any user inside the Channel to add his friends to Channel. EasyChat will send Event to all member's online SocketClient in *memberIds* and add them all to the SocketRoom. | | | |
| Remove People from Channel | number accountId number channelId | LeaveChannelResponseModel - *section 3.3.2* | to *channel\_participants* |
| This feature allows any user inside the Channel to remove anybody inside Channel. EasyChat will send Event to removed account's online SocketClient and also remove them from the SocketRoom. His messages persist, but a record in *channel\_participants* will be erased. | | | |
| Send Chat message | number channelId string content number replyTo attachmentFile | ChatMessageFromDatabase - *section 3.3.2* | to, from *channel\_messages* from *accounts* |
| This feature allows current user to send message to a specific Channel. After message was successfully saved, EasyChat announces an Event to the Channel. Attachments will be uploaded to AmazonS3. | | | |
| Delete Chat message | number id | ChannelMessage - *section 3.3.2* | to *channel\_message* |
| This feature allows current user to unsent his message. After message was successfully saved, EasyChat announces an Event to the Channel. | | | |
| Update Profile | avatarFile | Account - *section 3.3.2* | to *accounts* |
| This feature allows current user to update his profile. There is only avatar update for now. Avatar file will be uploaded to Amazon S3. | | | |

# System Designs

## Entity Relationship Diagram

Text

Description automatically generated with medium confidence

## Database Design

EasyChat database design was made as simplest as possible, neglecting all relationship among tables to fasten the performance. Hence, for the sake of execution time, I use pure SQL, no seed data, everything can be created by user at runtime.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| accounts | | | | |
| Field | Datatype | PK | NULL | Description |
| id | int | ● | ○ |  |
| username | varchar(256) | ○ |  |  |
| avatar\_url | varchar(512) | ○ | ● |  |
| last\_active | datetime | ○ | ● | Default: current system time |
| account\_friends | | | | |
| Field | Datatype | PK | NULL | Description |
| account\_id | int | ● | ○ |  |
| friend\_id | int | ● | ○ |  |
| is\_pending | bool | ○ | ● | Default: true |
| message | varchar(256) | ○ | ○ |  |
| added\_date | datetime | ○ | ● | Default: current system time |
| channels | | | | |
| Field | Datatype | PK | NULL | Description |
| id | int | ● | ○ |  |
| name | varchar(256) | ○ | ○ |  |
| avatar\_url | varchar(512) | ○ | ● |  |
| channel\_participants | | | | |
| Field | Datatype | PK | NULL | Description |
| channel\_id | int | ● | ○ |  |
| account\_id | int | ● | ○ |  |
| join\_date | datetime | ○ | ● | Default: current system time |
| is\_muted | bool | ○ | ● | Default: false |
| is\_disabled | bool | ○ | ● | Default: false |
| channel\_messages | | | | |
| Field | Datatype | PK | NULL | Description |
| id | bigint | ● | ○ |  |
| channel\_id | int | ○ | ○ |  |
| sender\_id | int | ○ | ○ |  |
| sent\_time | datetime | ○ | ● | Default: current system time |
| content | text | ○ | ● |  |
| file\_urls | text | ○ | ● | Array |
| file\_content\_type | varchar(16) | ○ | ● |  |
| is\_seen | text | ○ | ● | Default: false |
| is\_event | text | ○ | ● | Default: false |
| is\_deleted | text | ○ | ● | Default: false |
| reply\_to | bigint | ○ | ● |  |

## System functions design

### **3.3.1. Use-case diagrams & Sequence diagrams**

Belows are Use-case diagrams (UCD), Sequence diagrams (SD) correspond to each implemented function mentioned in **section 2.2,** drawn by *Draw.IO***:**

* **Mock authentication**

Diagram

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Fetch friends**

Diagram

Description automatically generated

*UCD*

A screenshot of a computer

Description automatically generated with medium confidence

*SD*

* **Fetch Pending Friends**

Diagram

Description automatically generated

*UCD*

A picture containing text, indoor

Description automatically generated

*SD*

* **Add Friend**

Graphical user interface, application

Description automatically generated

*UCD*

A screenshot of a computer

Description automatically generated with medium confidence

*SD*

* **Confirm friend**

Diagram

Description automatically generated

*UCD*

A picture containing graphical user interface

Description automatically generated

*SD*

* **Unfriend**

Graphical user interface, application, PowerPoint

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Create channel**

Diagram

Description automatically generated

*UCD*

A screenshot of a computer

Description automatically generated with medium confidence

*SD*

* **Fetch channels**

Diagram

Description automatically generated

*UCD*

A screenshot of a computer

Description automatically generated with medium confidence

*SD*

* **Fetch full channel**

Diagram

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Leave channel**

Graphical user interface, application, PowerPoint

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Update channel**

Diagram

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Fetch channel participants**

Diagram

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Add people to Channel**

Graphical user interface, diagram

Description automatically generated with medium confidence

*UCD*

Timeline

Description automatically generated

*SD*

* **Remove people from Channel**

Diagram

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Send Chat message**

A picture containing diagram

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

* **Delete Chat message**

Graphical user interface, text, application

Description automatically generated

*UCD*

Graphical user interface, diagram

Description automatically generated

*SD*

* **Update profile**

Graphical user interface, application

Description automatically generated

*UCD*

Diagram

Description automatically generated

*SD*

### **3.3.2. Class diagrams**

Developed in *Modular Architectural* style, EasyChat is a horizontally n-tier application, concluding *Modules*, *Data*, *Models*, *Utils*. While each package plays its significant role, sub-packages in *Modules* are pluggable. Such means, developing on one module will not damage others.

Graphical user interface, application

Description automatically generated

*EasyChat root package*

*Modules* package contains *Features* (System Functions) as indicated in **section 2.2**, which groups of them are organized in modules.

Graphical user interface, application

Description automatically generated

*app.modules package*

Accordingly, inside each module, there are

* *models* package: *Entities*, directly referable to a *Table* in *Database.*
* *features* package: *Feature* class that setup the *EventListener*, receiving a serializable *SocketMessage* which SocketIO parses client’s request data to.
* *Repository* class: this class handles the database interaction, absorbing the *AbstractRepository*’s functions from *app.data* package.
* Optional *Utilities* class: just some helpers used only for the module.

*Text

Description automatically generated*

*Data* package provides Hibernate setup and Repository facade.

Graphical user interface, application

Description automatically generated

*app.data package*

*Models* package has the custom entities that can be mapped to Hibernate queries.

Graphical user interface

Description automatically generated

*app.models package*

*Utils* package contains some singleton setup, Event constants and Tools.

Text

Description automatically generated

Graphical user interface

Description automatically generated

*app.utils package*

# Tasks sheet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Start time** | **Finish time** | **Implemented by** | **Evaluation (%)** |
| **Feasibility study phase** | | | | **10** |
| Choosing topic | 11/1/2021 | 11/1/2021 | Tung Pham Duc | 50 |
| Contributing ideas | 11/1/2021 | 11/1/2021 | Tung Pham Duc | 40 |
| Approval | 11/1/2021 | 11/1/2021 | Tung Pham Duc | 10 |
| **Requirement Analysis phase** | | | | **20** |
| Concept of operations | 11/2/2021 | 11/2/2021 | Tung Pham Duc | 20 |
| Listing functions & its definitions | 11/2/2021 | 11/3/2021 | Tung Pham Duc | 40 |
| Requirement tracer | 11/3/2021 | 11/3/2021 | Tung Pham Duc | 30 |
| Test plan | 11/4/2021 | 11/4/2021 | Tung Pham Duc | 10 |
| **Design phase** | | | | **25** |
| Technology election | 11/5/2021 | 11/5/2021 | Tung Pham Duc | 20 |
| Entity Relationship Diagram | 11/5/2021 | 11/5/2021 | Tung Pham Duc | 10 |
| Use-case diagrams | 11/5/2021 | 11/7/2021 | Tung Pham Duc | 20 |
| Sequence diagrams | 11/8/2021 | 11/11/2021 | Tung Pham Duc | 30 |
| Class diagrams | 11/11/2021 | 11/13/2021 | Tung Pham Duc | 20 |
| **Development phase (name of features)** | | | | **30** |
| **Please consult the UC Diagrams** | | | |
| Mock authentication | 11/14/2021 | 11/14/2021 | Tung Pham Duc | 8 |
| Fetch Friends | 11/15/2021 | 11/15/2021 | Tung Pham Duc | 3 |
| Fetch Pending Friends | 11/16/2021 | 11/16/2021 | Tung Pham Duc | 3 |
| Add Friend | 11/17/2021 | 11/17/2021 | Tung Pham Duc | 3 |
| Confirm Friend | 11/18/2021 | 11/18/2021 | Tung Pham Duc | 12 |
| Unfriend | 11/19/2021 | 11/19/2021 | Tung Pham Duc | 3 |
| Create Channel | 11/20/2021 | 11/20/2021 | Tung Pham Duc | 15 |
| Fetch Channels | 11/21/2021 | 11/21/2021 | Tung Pham Duc | 10 |
| Fetch Full Channel | 11/22/2021 | 11/22/2021 | Tung Pham Duc | 5 |
| Leave Channel | 11/23/2021 | 11/23/2021 | Tung Pham Duc | 3 |
| Update Channel | 11/24/2021 | 11/24/2021 | Tung Pham Duc | 5 |
| Fetch Channel Participants | 11/25/2021 | 11/25/2021 | Tung Pham Duc | 2 |
| Add People to Channel | 11/26/2021 | 11/26/2021 | Tung Pham Duc | 6 |
| Remove People from Channel | 11/27/2021 | 11/27/2021 | Tung Pham Duc | 3 |
| Send Chat message | 11/28/2021 | 11/28/2021 | Tung Pham Duc | 12 |
| Delete Chat message | 11/29/2021 | 11/29/2021 | Tung Pham Duc | 3 |
| Update Profile | 11/30/2021 | 11/30/2021 | Tung Pham Duc | 4 |
| **Testing phase** | | | | **10** |
| Testing application | 11/28/2021 | 11/29/2021 | Tung Pham Duc | 80 |
| Fixing bugs | 11/30/2021 | 11/30/2021 | Tung Pham Duc | 20 |
| **Deployment phase** | | | | **5** |
| Building application | 11/30/2021 | 11/30/2021 | Tung Pham Duc | 70 |
| Writing installation instruction | 11/30/2021 | 11/30/2021 | Tung Pham Duc | 30 |
|  |  |  |  | **100** |