Master Team Project WS2021
Milestone 1: HelpMeLearn

Master Team Project Winter 2021

Global Distributed Software Development

CEO & CTO:

Prof. Dr. Rainer Todtenhöfer

Team D

Hasib Iqbal (hasib.iqbal@informatik.hs-fulda.de)	:	Team Lead | Frontend Developer
Mohammad Rakibul Hasan	:	Frontend Lead
Mohammad Salman Haydar	:	Backend Lead
Talha Jahangiri Khan	:	Github Lead | Backend Developer
Chowdhury Amlan Barua	:	Cloud & Backend Developer

Revision	Date
Version 1.0	23 November 2021

23 November 2021

Table of Contents

Executive Summary

Nowadays, humanity is used to clicks, to the most convenient way to get to what the person wants. In the world of clicks, we want to try to make life easier for people both students and teachers. However, we have decided to focus on one of the most primary needs of everyone, which is a learning. HelpMeLearn is online tutoring platform where students and teachers get connected together.

HelpMeLearn is being developed exclusively for students and teachers from of Hochschule Fulda and San Francisco State University. In HelpMeLearn, we aim to help students to learn from experienced teachers. It does not matter if a student looking for some extra lectures on a specific subject to clear his/her concept or wants to know more about that course, we are here to support that student and do the hard job of finding a suitable teacher for him/her. In HelpMeLearn, users can search among thousands of courses. With the help of our unique matching system, we will find the most suitable teacher for a particular student who wants to learn. Students can instantly contact our teachers through our chat system and our private chat system ensures their security and ease of use.

Through HelpMeLearn, we offer one-to-one learning solutions for students of Hochschule Fulda and we connect learners with qualified, expert tutors online, on-demand, 24/7/361. We provide tutoring services in numerous academic subjects and test preparation areas in an engaging and uplifting learning environment. Our core philosophy is that when a learner needs help, the best way to get it is right away from an experienced teacher. Our mission is to help every learner first realize and then reach their full potential.

Personae & Main Use Cases

Here, the categories of users who are likely to use the application will be discussed, along with the use cases in which the application will be useful. The application usage will assume that all users using it have a stable internet connection and a PC/Laptop to allow for adequate viewing of the web application.

Personae

Type 1: Administrator

Administrator is responsible for maintaining the application. This type of user has full access to the application & he must verify posts of the tutors.

Type 2: Teacher

Personal who is qualified or expert in a field & wants to instruct people on that field.

Type 3: Student

Personal who thinks he needs to learn about some topic from an expert.

Type 4: Guest

Personal who can see limited offerings in the web application but is not allowed to use the platform.

Main Use Cases

Use Case 1: A student wants to learn about a new topic.

A student has just started his semester in Hochschule Fulda. After some weeks, he thinks he should have some extra lectures on a specific subject to clear his concept. So, he thinks of going for a student tutor, & search for one in the tutoring service platform. After registering, he can see all the tutoring offering that is available & verified by administrator. He can then contact to the tutor that matches his expectation.

Use Case 2: A student who knows about a topic but wants to know more from an expert.

A student is studying in Hochschule Fulda. He has completed some courses but one of the courses seems interesting to him. He wants to know more about that course & thinks that he can look for someone who is expert in that topic. He then registered as a student in the tutoring service platform & searched for an expert tutor in that field. He then messaged that tutor, & discussed how the tutor wants to give some lectures.

Use Case 3: A graduated student is looking for a tutor to brush up his knowledge.

A student is graduated from Hochschule Fulda. He is now working in some industry for 3 years. For a new project, he needs to work on something new that he has little knowledge. So, he decided to know about that topic from someone. He then registered in the tutoring service platform & searched for an expert tutor in that field. He then messaged that tutor, & states his expectation. With the help of the platform, he got in touch with someone who can meet his expectation.

Use Case 4: An outgoing student has gathered expertise in some specific field & thought he can share his expertise

A student who is nearly at the end of graduation has gathered an expertise in numerous fields. He thought he can share his expertise with another student. He recorded some demo lectures on some topics and register as a tutor on the tutoring service platform. Then he posts a position for tutoring with some of his demo lectures. It gets reviewed by the administrator & then get published in the platform. Students can now see his offering & some students messaged him to have lecture from him.

List of main data items and entities

- **Users:**
  - **Student** -> Any user interested in finding coaching for subjects.
  - **Tutor** -> Users that are experienced in teaching students and offering coaching to interested students to help in subjects.
  - **Admin** -> Super user that helps maintain order on the website. Such type of user has full access to approve / decline or revoke posts (content) on website. Delete users that violate rules. Has full privilege over the website.
  - **Unregistered users (guest) ->** Users that can access the website without logging in to a specific account. Limited actions.

- **Comments:** Comments on the profile of the tutor related to feedback of services.

- **Images:** All the images including the tutor profile picture, other related content.

- **CV:** Entity will contain the information of tutors including qualifications, subjects taught, background, experience.

- **Messages:** Text exchanged between students and tutors.

- **Preferences:** List of values defined by users(students) for their ideal match to tutors.

Initial List of functional requirements

No	Description
1	There will be four user categories e.g., admin, tutor, and students and unregistered users
2	Any unregistered user (Tutor or Student) from Fulda or SFSU should be able to register themselves
3	Admin should have the ability to approve/disapprove any tutoring post from the user (tutor)
4	Tutor should be able to post his/her tutoring add on the platform with necessary information. Which includes image, CV, available time slot, subject etc.
5	Students should be able to search on the system to find their desirable tutor by several topics like subject name, level (undergraduate/masters) etc.
6	Student and tutor should be able to communicate with the live chat system in our platform
7	Admin should be able to delete or request for update of any incomplete or inappropriate items or users
8	Students can set preferences for tutors to have a good match with required preferences.
9	Poll can be made for voting for required course by tutors that are not available for tutoring.
10	Students can comment as feedback for their services on tutor profile.

List of non-functional requirements

- Application shall be developed, tested, and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team, but all tools and servers have to be approved by class CTO).

- Application shall be optimized for standard desktop/laptop browsers e.g., must render correctly on the two latest versions of two major browsers.
- All or selected application functions must render well on mobile devices.
- Data shall be stored in the database on the team's deployment server.
- No more than 50 concurrent users shall be accessing the application at any time.
- Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users.
- The language used shall be English (no localization needed).
- Application shall be very easy to use and intuitive.
- Application should follow established architecture patterns.
- Application code and its repository shall be easy to inspect and maintain.
- Google analytics shall be used (optional).
- No e-mail clients shall be allowed. Interested users can only message to sellers via in-site messaging.
- Pay functionality, if any (e.g., paying for goods and services) shall not be implemented nor simulated in UI.
- Site security: basic best practices shall be applied (as covered in the class) for main data items.
- Media formats shall be standard as used in the market today.
- Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.
- The application UI (WWW and mobile) shall prominently display the following exact text on all pages "Fulda University Software Engineering Summer 2021. For Demonstration Only" at the top of the WWW page. (Important to not confuse this with a real application).

Competitive Analysis

Feature	Finde	SolutionInn	Chegg	Khan Academy
Search courses with filters.	Yes	Yes	Yes	Yes
Send message to teachers	Yes	Yes	Yes	Yes
Register as a student or Teacher.	Yes	Yes	Yes	Yes
Poll (Vote) based course creation	Yes	No	No	No
Information about Upcoming courses	Yes	No	No	Yes
Teacher-Student blog	No	No	Yes	Yes
Online Video Lessons	No	No	Yes	Yes

The internet nowadays is overpopulated with online tutoring websites, but we feel that our website will make a difference in the market. Having listed the main features of our competitors, we noticed that apart from the main basic features that are almost equally amongst competition, there are some subtle, yet effective features offered among them. For example, only our website offers vote-based course creation which means Teacher can create a poll where students can vote on which topics they are interested to learn. According to Poll, teachers will understand on which course or topic students are struggling with.

High-level System Architecture & Technologies

Technology Stack

- **Server Host:** Amazon AWS and Google Cloud [1vCPU 2 GB RAM]
- **Operating System:** Ubuntu Server 20.04 LTS
- **Server Database:** MySQL V8.0.27
- **Web Server:** NGINX V1.20.1 (Ubuntu)
- **Server-Side Technologies:** NodeJS
- **Front-End Technologies:** HTML, CSS, JavaScript

Additional Technologies

- **IDE:** Visual Studio Code
- **Task Scheduling:** Microsoft Teams
- **Back-End Library & Frameworks:** Express JS V4.17.1
- **Front-End Library & Frameworks:** React V17.0.2, Bootstrap V5.1.3

Team and roles

Our team is consisting of five talented software engineers. Based on two meetings that we had during the first week, we got to know each other more and know our expertise, and decide which roles should be assigned to whom.

Here is a summary of the team members and their tasks during this project:

Team Member	Role
Hasib Iqbal	Team Lead | Frontend Developer
Mohammad Rakibul Hasan	Frontend Lead
Mohammad Salman Haydar	Backend Lead
Talha Jahangiri Khan	Github Master | Backend Developer
Chowdhury Amlan Barua	Cloud & Backend Developer

Checklist

Task	Status
Team found a time slot to meet (online) outside of the class	Done
GitHub master chosen	Done
Team decided and agreed together on using the listed SW tools and deployment server	Done
Team ready and able to use the chosen back and Frontend Frameworks and those who need to learn are working on learning and practicing.	Done
Team lead ensured that all team members read the final M1 and agree/understand it before submission.	Done
GitHub organized as discussed in class	Done

Master Team Project WS2021

Milestone 2: HelpMeLearn

Master Team Project Winter 2021

Global Distributed Software Development

CEO & CTO:

Prof. Dr. Rainer Todtenhöfer

Team 04

Hasib Iqbal (hasib.iqbal@informatik.hs-fulda.de)	:	Team Lead | Frontend Developer
Mohammad Rakibul Hasan	:	Frontend Lead
Rohat Sagar Urif Sonu	:	Frontend Developer
Mohammad Salman Haydar	:	Backend Lead
Talha Jahangiri Khan	:	Github Lead | Backend Developer
Chowdhury Amlan Barua	:	Cloud & Backend Developer
Nisha Devi	:	Backend Developer

Revision	Date
Version 1.0	7 December 2021
Version 2.0	17 December 2021

17 December 2021

Table of Contents

Functional Requirements:

PRIORITY 1: MUST HAVE

Admin Users:
View of pending posts: Admin users should be able to view a list of posts that are pending approval.
Add new course: Admin users should be able to add new courses from any department.
Block User: Admin users can block any user (Students & Tutors) who has posted an inappropriate post on the site.
Block posted Post: Admin users can block any post after being published.

Unregistered Users:
Register: Unregistered users should be able to register to the website when they register using the Hochschule Fulda email address.
View Post list: Guest users can view the list of posts posted on the website with limited information.

Registered Users:
Login: Registered users should be able to log in using their correct login credentials.

Tutors:
Add a Post: Registered tutors should be able to post his/her tutoring ad with the necessary information. Which includes image, CV, available time slot, and subject.
View Post lists: Registered tutors should be able to view the list of posts they posted.
Send Message: Registered Tutors can reply to messages to the students.
View Message Details: The tutor can read the conversation thread with another user or students.

Students:
View Post list: Users can view the list of posts posted on the website.
Search Post: Users can search posts by their subject, level, price & gender.
Sort Post: Users can sort published or search result items by published/posted date and review in ascending or descending order.
View details of Post: Users can view the details of published posts like User Name, date, etc.
Comment on Post: Users can comment or give feedback on the tutor post.
Send Message: Students can message the tutors.

PRIORITY 2: DESIRED

Tutors:
Include video in post: Tutors can add a video in post.

Students:
Set Preference: Users can set their preferences based on topic, time, and level.
Recommend tutor: Based on preferences, the website will recommend tutors to the registered students.

Voice searching: Users can search for tutors using voice commands.

PRIORITY 3: OPPORTUNISTIC

Suggest possible meeting schedule: Suggest a possible available time for a meeting between student & tutor based on their available time.

Admin Users:

Bulk Approve Posts or Users: Admin can approve the bulk of posts or users that are pending approval.

Students:

Add a Post: Students can add a post for their preferences for a tutor.

Main Data Items and Entities

In this section, a general description of data and entities will be discussed. The description of data is similar to milestone 1 with some modifications. The main entities involved in the project as listed as:

- **User**
    1. User ID
    2. Username: Unique Identifier of user-created at first login
    3. First Name
    4. Last Name
    5. Type: Indicates type of user (tutor, student, admin)
    6. Email
    7. Password: password is stored as a hash for security reason
    8. Status: Indicates the status of the user (active, suspended)
    9. UserType:ENUM

- **POST**
    1. Post ID
    2. Title
    3. Description (description including no of rooms, space, area)
    4. Schedule
    5. Creator Id (Id of the user who created the post)
    6. Created At (Timestamp)
    7. Updated At (Timestamp)
    8. Status: ENUM

- **CHAT**
    1. Chat ID
    2. TutorID
    3. StudenrID

4. Created At (Timestamp)

- **TEXT**
  1. Text ID
  2. Chat ID

- **Document**
  1. DocumentID
  2. PostID

- **Comment**
  1. UserID
  2. PostID
  3. Created At (Timestamp)

UI Mockups and Storyboards

Attached at the end of the document, to not disturb the overall layout flow of the Milestone 2 document.

High-level Architecture, Database Organization

Here, the database schema/organization such as its DB tables and items in each table will be discussed. Additionally, initial details regarding media storage and search/filter implementation for database items will be discussed as well.

Database Schema:

This is the initial, high-level description of the database schema and its tables. Additional tables might be created, and current tables may change based on future decisions and feature implementations.

The schema will include four tables detailed as follows:

- **USER**: This is the table that contains all relevant user information:

Column	Description
UserId	Unique identifier. Will be used as primary key
UserName	Name of User
Type	Default is guest but can be changed by student, tutor or admin
Email	Email of user
Status	Shows status of the user as either “active” or ‘suspended’
UserType(ENUM)	Name of each type of user such as: Student, Tutor, Admin

- **POST**: This is the table that holds data of a post, where events and announcements by a Tutor are posted.

Column	Description
PostID	Unique identifier. Will be used as primary key
Title	Title of the post
TutorID	ID of the user who created the post
CreatedOn	Date when the post was created
Subject	Title of the post
Schedule	Post description
Status (ENUM)	Status of the post
Language	Prefered Language

- **TEXT:** This table will hold that of text message of chat between Tutor and Student

Column	Description
TextID	ID of each text
ChatID	ID of chat between Tutor and Student

- **CHAT:** This table will hold the data of Tutors and Students.

Column	Description
ChatID	ID of each chat
TutorID	ID of Tutor
StudentID	ID of Student

- **DOCUMENTS:**

Column	Description
DocumentID	ID of document
POSTID	ID of Post related to the Document

- **COMMENTS:**

Column	Description
PostID	ID of a Post
UserID	ID of a user who comments
Created At	Time of Comment

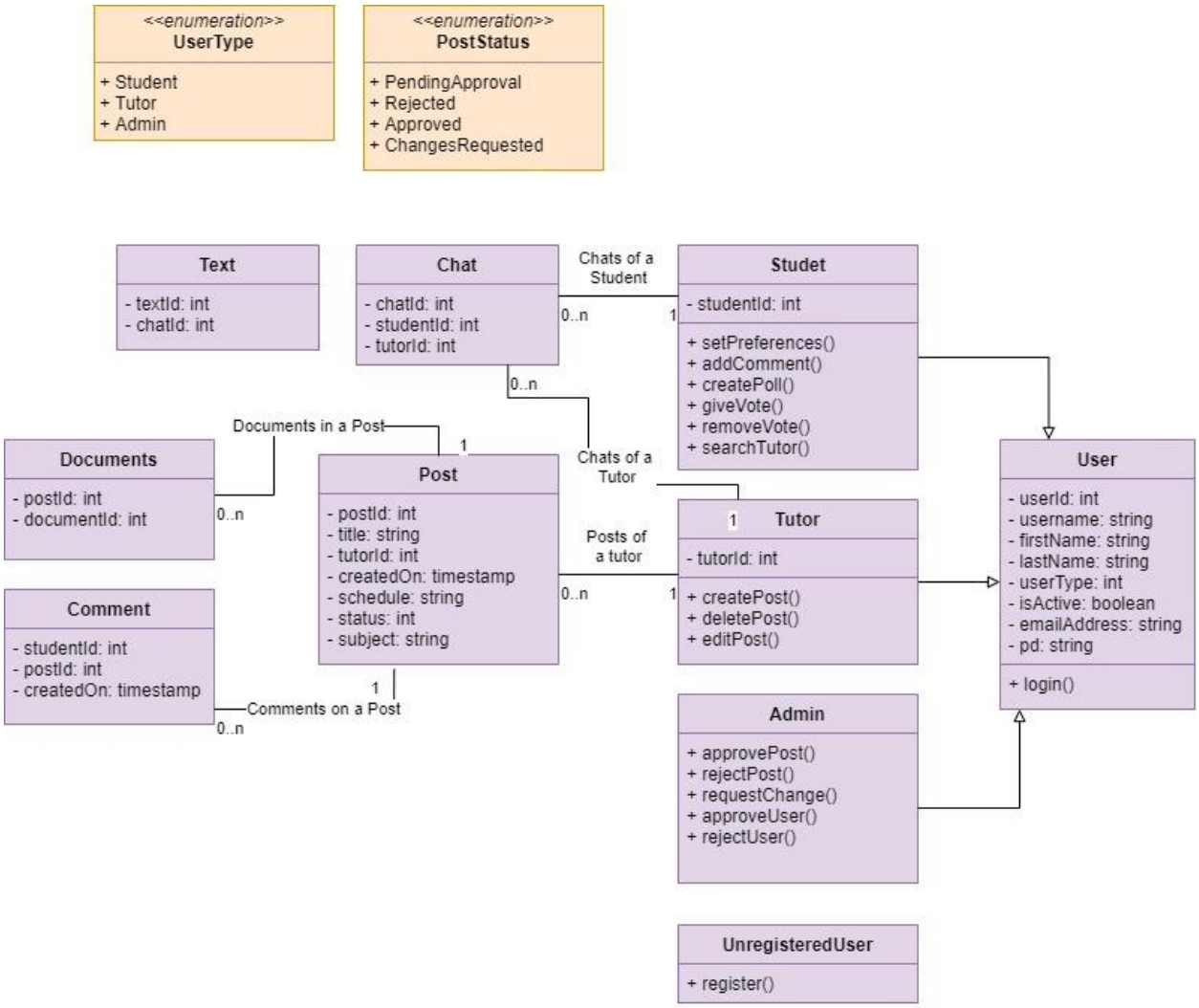
Media Storage:

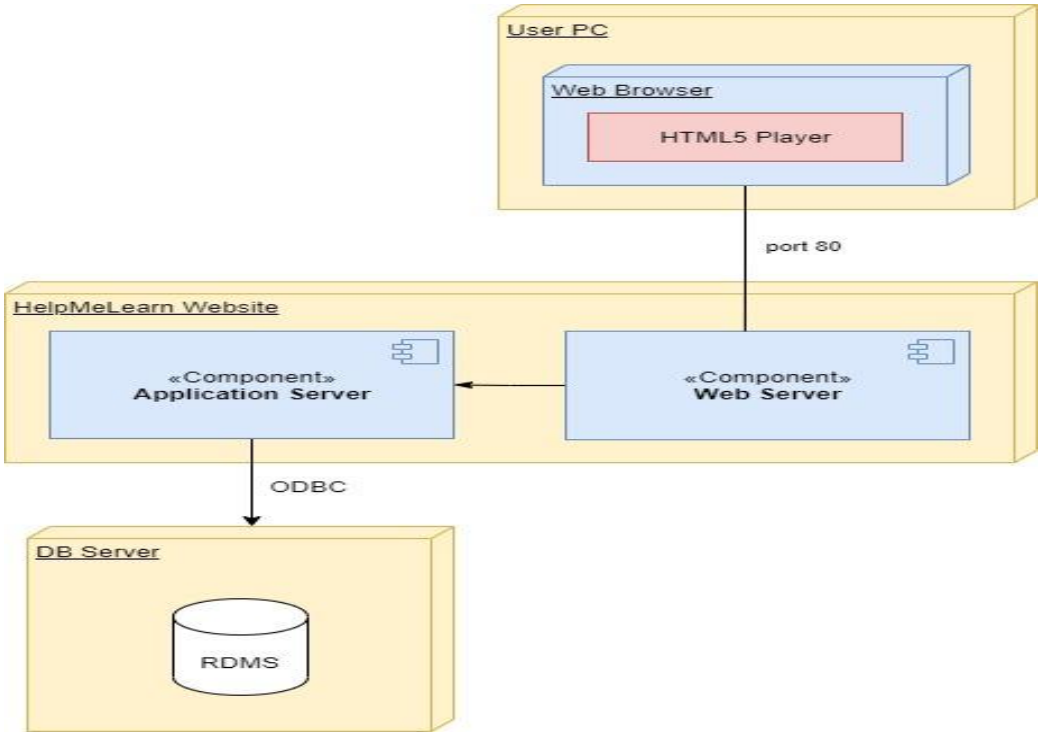
For media storage, File storage will be used as it makes organization of data easier. We will create a folder within our project for storing media files. Then store the relative path of the media directory in the MySQL database. The expected size of one picture is between 700 kbs to 3MBs. The expected size of the relative path is 300 characters.

Search/Filter Implementation:

For the implementation of the search/filter: The TutorName, Department, and SubjectType will be available as a drop-down for the user to select one to be applied to the query. “SubjectType” and the “Title of Post” are the searchable terns which can be derived from POST table. In the case of searching for similar words, the LIKE operator will be used, which will look for similarities between user input and the columns “title” and “username”. To date by date or price, the terms “timestamp” or “price” will be sorted either in ascending or descending order using the “ORDER BY” and “ASC” or “DESC” operations.

High-Level UML Diagrams




User PC
Web Browser
HTML5 Player
port 80
HelpMeLearn Website
«Component»
Application Server
«Component»
Web Server
ODBC
DB Server
RDMS

Key Risks

- **Skills Risk:**

We have software development experience on different software platforms in our bachelor's programs. Furthermore, most of us have little experience developing tutoring websites. While developing this course project, we as a team may lack technical skills for individuals to cover all the aspects of the project. Eventually, team members will learn the skills, this learning may include programming languages, frameworks, and other skills related to software development practices. We may have online sessions to cover up these issues and team members who have specific experience on specific technology will help others and vice versa.

- **Schedule Risk:**

Project estimations can be wrong or incorrect when project tasks and scheduled releases are not analyzed properly. Schedule risks mainly affect a project and may lead to project failure. Wrong deadline estimations, inappropriate tracking of resources like staff, systems, and skills of individuals, and failure to identify problems in the project can lead to the delay of the project.

To overcome these risks, we will use planning documents, such as specifications and project plans, and perform a detailed task analysis of the work to be performed so that we can reduce the critical paths and dependencies available.

- **Technical Risk:**

Technical risks may lead to failure of functionality or performance in the production environment. This may occur if we use deprecated frameworks/plugins or any dependencies which need to be updated all the time to maintain the consistency of the project. We will use updated frameworks, API, plugins, or any other dependencies.

- **Teamwork Risk:**

Shared values and coordination of expertise are important factors for team leaders to consider achieving high-quality software teamwork. Since team members are sharing most of the responsibilities to deliver within deadlines, some individuals may contribute less than others. This may create a negative perception among team members. Also, sometimes miscommunication may happen among team members. All these issues may lead to some negative perceptions that can make the team less effective. We should clearly mention the task responsibilities and accountability for individual contributions to maximize the group effort.

- **Legal/Content Risk:**

For the software industry, legal risk management is a growing concern. In some cases, it can be a serious threat to the commercial and financial success of software systems. As a student, sometimes we may not find guidance on legal assurance, as it is not covered in the software best practice frameworks and international standards. Especially using any public APIs, plugin, or software snippet may lead to copyright allegations and claims. We will be working with the services of open source communities wherever possible and give proper credits and acknowledgments wherever needed.

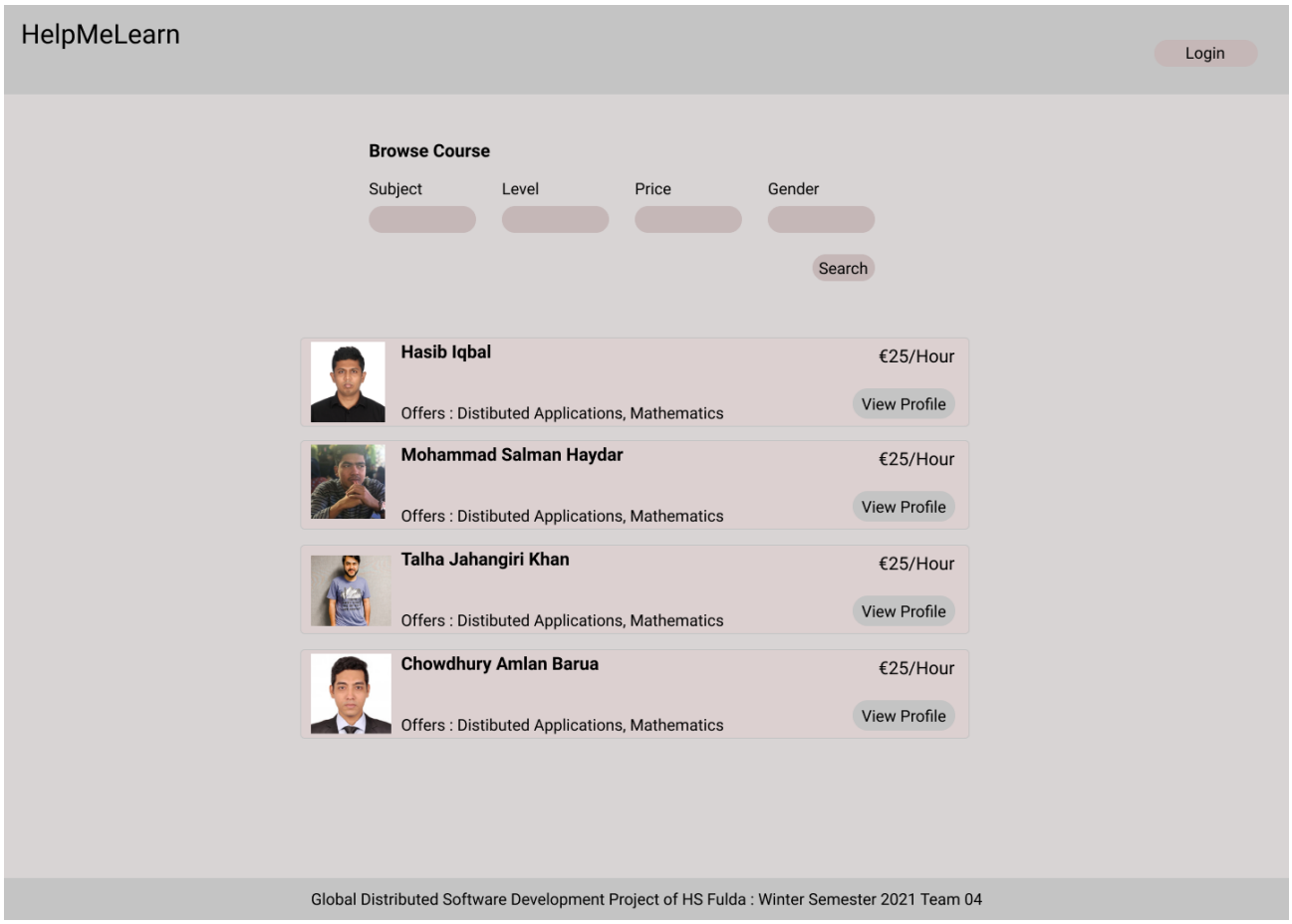
Project Management

For project management, we are using Microsoft Teams. There we can use ‘Tasks’ for maintaining our due features. Also, we use this platform to do weekly meetings to discuss the project.

UI Mockups and Storyboards

Figma Prototype Link:

https://www.figma.com/proto/LyS2L4GtfQQcopDyKmPNGk/HelpMeLearn?node-id=70%3A2&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=70%3A2



HelpMeLearn

Log in

Email

Password

Login

Forget Password?

No account? Register Instead

HelpMeLearn

Registration

Name :

Gender :

Email :

Password :

Confirm Password :

User Type :

Create

Global Distributed Software Development Project of HS Fulda : Winter Semester 2021 Team 04

HelpMeLearn

Hello, Rakib.

Browse Course

Subject | Level | Price | Gender

Search

Hasib Iqbal €25/Hour

Offers : Distibuted Applications, Mathematics

View Profile

Mohammad Salman Haydar €25/Hour

Offers : Distibuted Applications, Mathematics

View Profile

Talha Jahangiri Khan €25/Hour

Offers : Distibuted Applications, Mathematics

View Profile

Chowdhury Amlan Barua €25/Hour

Offers : Distibuted Applications, Mathematics

View Profile

HelpMeLearn

Hasib Iqbal

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Arcu felis bibendum ut tristique et egestas quis ipsum. Urna id volutpat lacus laoreet non curabitur gravida. Egestas fringilla phasellus faucibus scelerisque.

Offered Courses

Distributed Application €30/Hour

Details

Mathematics €25/Hour

Details

Qualifications

Subject	Qualification Level	Grade
Distributed Application	Bachelor	A+
Mathematics	Bachelor	A+

Reviews

Rakib Hasan

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod.

Rohat

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod.

Message

HelpMeLearn

Hasib Iqbal

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Arcu felis bibendum ut tristique et egestas quis ipsum. Urna id volutpat lacus laoreet non curabitur gravida. Egestas fringilla phasellus faucibus scelerisque.

Offered Courses

Distributed Application €30/Hour Details

Mathematics €25/Hour Details

Qualifications

Subject	Qualification Level	Grade
Distributed Application	Bachelor	A+
Mathematics	Bachelor	A+

Reviews

Rakib Hasan

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod.

Rohat

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod.

Hello, I have checked your profile. I'm interested to learn Distributed Application from you.

|Aa Send

Message

Global Distributed Software Development Project of HS Fulda : Winter Semester 2021 Team 04

HelpMeLearn

Edit Profile

Hasib Iqbal

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Arcu felis bibendum ut tristique et egestas quis ipsum. Urna id volutpat lacus laoreet non curabitur gravida. Egestas fringilla phasellus faucibus scelerisque.

Offered Courses

Distributed Application €30/Hour Details

Mathematics €25/Hour Details

Qualifications

Subject	Qualification Level	Grade
Distributed Application	Bachelor	A+
Mathematics	Bachelor	A+

Reviews

Rakib Hasan

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod.

Rohat

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod.

Global Distributed Software Development Project of HS Fulda : Winter Semester 2021 Team 04

HelpMeLearn

Change Picture

Hasib Iqbal

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Arcu felis bibendum ut tristique et egestas quis ipsum. Urna id volutpat lacus laoreet non curabitur gravida. Egestas fringilla phasellus faucibus scelerisque.

Save | Edit

Offered Courses Add New Course

Distributed Application	€30/Hour Details
Mathematics	€25/Hour Details

Curriculum Vitae Add New Course

Hasib's CV.pdf

Qualifications Add Qualification

Subject	Qualification Level	Grade
Distributed Application	Bachelor	A+
Mathematics	Bachelor	A+

HelpMeLearn

Add New Course

Department

Subject

Request to add subject

Description

Per Hour Fee

Years of Experience

Available Time

Request for Approval

HelpMeLearn

Add Qualification

Subject

Qualification

Grade

Save

HelpMeLearn

Tutor List Student List

Welcome, Admin.

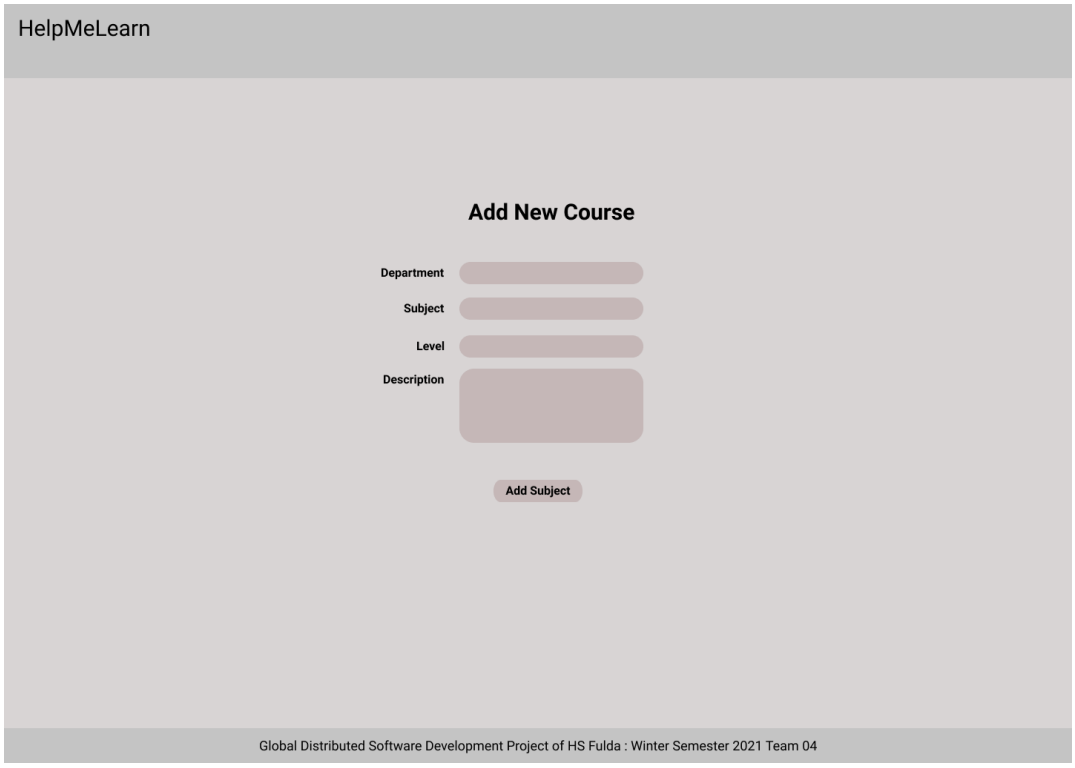
Pending Tutor Requests

Tutor Name	Subject Name			
Rakib Hasan	Mathematics	View	Approve	Reject
Salman Haydar	Discrete Mathematics	View	Approve	Reject

Pending Course Requests

Add New Subject

Department	Subject			
Applied Computer Science	Programming 1	View	Approve	Reject
Sociology	Sociology 2	View	Approve	Reject


HelpMeLearn
Add New Course
Department
Subject
Level
Description
Add Subject
Global Distributed Software Development Project of HS Fulda : Winter Semester 2021 Team 04

Master Team Project WS2021

Milestone 3 Part 1 Review Summary: HelpMeLearn

Master Team Project Winter 2021

Global Distributed Software Development

CEO & CTO:

Prof. Dr. Rainer Todtenhöfer

Team 04

Hasib Iqbal (hasib.iqbal@informatik.hs-fulda.de)	:	Team Lead | Frontend Developer
Mohammad Rakibul Hasan	:	Frontend Lead
Rohat Sagar Urif Sonu	:	Frontend Developer
Mohammad Salman Haydar	:	Backend Lead
Talha Jahangiri Khan	:	Github Lead | Backend Developer
Chowdhury Amlan Barua	:	Cloud & Backend Developer
Nisha Devi	:	Backend Developer

Revision	Date
Version 1.0	25 January 2022

25 January 2022

Summary of feedback and tasks to do:

The status of the project was good, the professor checked our project and most of the functionalities were done and worked properly as discussed before and during writing our functional requirements and P1 functions.
The professor checked the search functionality and saw that we get the proper result based on our search and if we use the filtering, we will get a result based on the search criteria we put. About data upload, the professor checked that the user could upload a post including all the data with a picture and this post will be reviewed by the admin to be later approved or rejected based on the website policy that this functionality also works properly. Regarding the Admin dashboard, the professor also checks that the admin can review the posts and based on our website policy reject the post or approve it. About the user dashboard, the user can see his posts. The UI responsiveness of our website was also done and reviewed by the professor. About our Website performance, everything was loading in a proper and in a fast way and there was no crashing. What was left were chat functionality which is one of our P1 functions.

List of tasks the team chose to focus on and implement for final delivery

Our focus was from the beginning to deliver the P1 functions as we wrote in the M2 document, so we focused first on implementing the last P1 functions which are Chat functionality and we were discussing on implementing any new or interesting function like adding a voting poll for analyzing user interest on a subject.

Master Team Project WS2021

Milestone 4: HelpMeLearn

Master Team Project Winter 2021

Global Distributed Software Development

CEO & CTO:

Prof. Dr. Rainer Todtenhöfer

Team 04

Hasib Iqbal (hasib.iqbal@informatik.hs-fulda.de)	:	Team Lead | Frontend Developer
Mohammad Rakibul Hasan	:	Frontend Lead
Rohat Sagar Urif Sonu	:	Frontend Developer
Mohammad Salman Haydar	:	Backend Lead
Talha Jahangiri Khan	:	Github Lead | Backend Developer
Chowdhury Amlan Barua	:	Cloud & Backend Developer
Nisha Devi	:	Backend Developer

Revision	Date
Version 1.0	11 March 2022

11 March 2022

Table of Contents

Product Summary

Motivation & Importance

Humanity has become accustomed to clicks as the most expedient method of obtaining what one desires. In this age of clicks, we want to make life easy for students and teachers alike. However, we have chosen to focus on one of everyone's most basic requirements, which is learning. HelpMeLearn is a web-based tutoring service that connects students and teachers.

Functions & Services

Our product aims to establish a communication bridge between tutor and students. From the perspective of students, they can search for tutors for their desired subjects using our platform and also can communicate with them. Our platform provides real time chat facilities which can help the students to get connected with the tutors instantly. They can also share their experience about tutors through our review system. From the perspective of the tutor, they can upload their CV, qualification and show their skills through our tutor profile page to get the interests of the students. They can mention their hourly salary expectations, reply to messages from the students using our platform. Also, every change made by the tutor in their timeline is reviewed and approved by the admin before going live to maintain the standard of our platform. Our platform also provides a guest page where a guest user can search for their desired subjects and tutors without registering. The only catch here is that they cannot communicate with the tutor. Our searching page also provides facilities about filtering different categories like by subject, gender, etc.

Reasons to use our HelpMeLearn Application

Using our HelpMeLearn application for learning will make students and teachers connect in the most efficient way without having to pay a great sum of money for subscription and we are ensuring a safe platform for both users. If you are a student or teacher, it creates a safe platform for a teacher and a student to connect as the opportunity to offer courses by a teacher is endless and students can easily compare the offered courses.

Major Committed Functions:

1. Search courses

- Allowing/Enabling the students to find certain approved courses that satisfy / fulfill a criterion.

2. Filter searched courses

- Specific attributes a student can use to refine the search results or course. e.g., by gender, price range etc.

3. Add /View Reviews

- Allowing/Enabling the students to view reviews of teachers that can help students find suitable teachers

4. Registration

- Allowing the students and teachers to make a new account profile for themselves.

5. Login

- Allowing the students and teachers to log in to their account or dashboard.

6. Post Courses

- Allowing the approved teachers to add new courses from their profiles.

7. Messaging

- Allowing the students and teachers to communicate with each other through messages and also enables them to see the previous message history as well from their dashboard.

8. Approve/Reject courses / teacher information

- Allowing the admins to approve / reject different pending courses or personal information of teachers from their dashboard.

Usability Test Plan

This document describes a test plan for conducting a usability test during the development of HelpMeLearn. The goals of usability testing include establishing a baseline of user performance, establishing, and validating user performance measures, and identifying potential design concerns to be addressed to improve the efficiency, productivity, and end user satisfaction.

Test objectives

The objective of this testing is to get the user feedback and user experience about searching for a specific tutor using searched keywords, subject filter or choosing specific subject level.

Test background and setup

- **System setup**

  HelpMeLearn website have been published on AWS to be tested by participants using the following technologies

  • Operating System: Ubuntu 18.04 Server
  • Database: MySQL v: 8.0
  • Web Server: NGINX 1.18
  • Server-Side Language: Node.js

- **Starting point**

  Participants will test HelpMeLearn website using their laptops or smartphones, and will focus on the search functionality in the home page

- **The intended users**

  The test will be conducted with a targeted category of users, which are the students and tutors of Fulda university. Therefore, we will ask for users of the university to test the website

- **URL of the system to be tested**

  HelpMeLearn (the link may change due to redeploying)

- **What is to be measured**
  We will be focusing on user satisfaction

Usability Testing Phase:

<u>Test Participant 1:</u>

Phase 1: Screening and Pretest:

Q1. How old are you?
24

Q2. What is the highest level of education you've completed?
Bachelor's Degree

Q3. What is your current occupation?
Student(Pursuing Masters)

Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for a tutor online?
5

Q5. When was the last time you looked for a tutoring service online?
In 2020

Phase: 2 Usability Task description

Task 1: Imagine that you are facing difficulty in studying statistics and want to get help from an expert.

Task 2: Imagine that your exams are near and facing difficulty in solving exercises of economics

Task 3: Imagine you are looking for a tutor in fulda and worried about feedback from other students.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching for a tutor?
Good

Q2. How simple and clean was the interface?
It is very simple.

Q3. Can you tell me what you think about filtering the tutor by subject name and level?
I tried multiple times and it worked without any issues.

Q4. How was your experience looking for tutor qualifications and feedback?
Simple and clear.

Sl. No	Task Objective	Scale 0/5 (0-OK, 5-Good)
Task 1	Search for a tutor online by subject name	4/5
Task 2	Search for a tutor online by subject level	4/5
Task 3	Looking for tutor feedback	3/5

Test Participant 2:

Phase 1: Screening and Pretest:

Q1. How old are you?
19

Q2. What is the highest level of education you've completed?
College

Q3. What is your current occupation?
Student(Pursuing Bachelors)

Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for a tutor online?
5

Q5. When was the last time you looked for a tutoring service online?
In 2020

Phase: 2 Usability Task description

Task 1: Imagine that you are facing difficulty in studying statistics and want to get help from an expert.

Task 2: Imagine that your exams are near and facing difficulty in solving exercises of economics

Task 3: Imagine you are looking for a tutor in fulda and worried about feedback from other students.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching for a tutor?
Good and simple.

Q2. How simple and clean was the interface?
It is very simple and easy to use.

Q3. Can you tell me what you think about filtering the tutor by subject name and level?
Filters are clear and easy to use.

Q4. How was your experience looking for tutor qualifications and feedback?
Clearly mentioned.

Sl. No	Task Objective	Scale 0/5 (0-OK, 5-Good)
Task 1	Search for a tutor online by subject name	4/5
Task 2	Search for a tutor online by subject level	3/5
Task 3	Looking for tutor feedback	4/5

Test Participant 3:

Phase 1: Screening and Pretest:

Q1. How old are you?
21

Q2. What is the highest level of education you've completed?
University

Q3. What is your current occupation?
Student(Pursuing Bachelors in Natural Science)

Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for a tutor online?
4

Q5. When was the last time you looked for a tutoring service online?
In 2019

Phase: 2 Usability Task description

Task 1: Imagine that you are facing difficulty in studying statistics and want to get help from an expert.

Task 2: Imagine that your exams are near and facing difficulty in solving exercises of economics

Task 3: Imagine you are looking for a tutor in fulda and worried about feedback from other students.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching for a tutor?
Great and amazing.

Q2. How simple and clean was the interface?
Interface is very easy to use. And I did not have any issue adjusting to the interface.

Q3. Can you tell me what you think about filtering the tutor by subject name and level?
Filters are clear and easy to use. However, I would appreciate more filtering options.

Q4. How was your experience looking for tutor qualifications and feedback?
Clearly mentioned.

Sl. No	Task Objective	Scale 0/5 (0-OK, 5-Good)
Task 1	Search for a tutor online by subject name	4/5
Task 2	Search for a tutor online by subject level	4/5
Task 3	Looking for tutor feedback	3/5

QA Test Plan

Test Objective:

The test objectives are to verify the functionality of website HelpMeLearn and to ensure that the software is as per business requirements by identifying any bugs or issues and fixing them before release.

Hardware and Software setup:

Laptop or smartphone using Edge, Google Chrome or Firefox browser.

Feature to be tested:

- Search By Subject Name
- Search By Subject Level
- Search By Tutor Gender

Test Case Id & Browser	Test Title	Test Description	Test Input	Expected Result	Test Result
TC_1 (Chrome, Firefox)	View all tutors	Students are able to view all tutors in a list view.	Login with valid credentials. Students will be navigated to the Search Page	The Search Page displays all approved tutors.	PASS
TC_2 (Chrome, Firefox)	Filter Tutors by subject name	Students are able to filter tutors by subject name.	Input '**Math101**' on **SubjectName** text field and click on the **Search** button	Students are able to see all approved tutors who teach the **Math101** subject.	PASS
TC_3 (Chrome, Firefox)	Filter Tutors by subject level.	Students are able to filter tutors by subject level.	Select **'Master'** in the subject level drop down field and click on the **Search** button.	Students are able to see all approved tutors who teach at least one subject with a master's level.	PASS
TC_4 (Chrome, Firefox)	Filter Tutors by gender.	Students are able to filter tutors by gender.	Select **'Male'** in the **Gender** drop down field and click on the **Search** button.	Students are able to see all approved tutors whose gender is male.	PASS

TC_5 (Chrome, Firefox)	Filter Tutors by subject Name, level, and gender.	Students are able to filter tutors by subject name, level and gender.	Input **'Math101'** in the Subject Name Field, select **'Master'** in the subject level drop down field, select **'Male'** in the gender drop down field and click on the **Search** button.	Students are able to see all approved tutors who teach SubjectName 'Math101' with subject level 'Master' and gender is male.	PASS

Code Review

Developer	Rohat Sagar Urif Sonu
Reviewer	Nisha Devi

File: socketIO.js

socketIO.js M

C: > git > tutoringServiceGDSD > server > socketIO > socketIO.js > <unknown> > exports > io.on("connection") callback > socket.on("connectUser") callback

```
33  io.on("connection", (socket) => {
34    /* Review: Remove console.log and add try and catch and log the exception if occur. */
35    console.log("client connected");
36
37    socket.on("connectUser", async (payload) => {
38      const { userId } = payload;
39
40      if (userId === undefined || (await fetchUser(userId)) === undefined)
41        return;
42
43      addConnectedUser(userId, socket.id);
44
45      // emit previous texts
46      socket.emit("userTextsFetched", await fetchUserTexts(userId));
47    });
48
49    socket.on("sendText", async (payload) => {
50      const { from, to, text } = payload;
51
52      if (from === undefined || to === undefined) return;
53
54      const fromUser = await fetchUser(from);
55
56      const toUser = await fetchUser(to);
57
58      if (fromUser === undefined || toUser === undefined) return;
```

socketIO.js M

C: > git > tutoringServiceGDSD > server > socketIO > socketIO.js > <unknown> > exports > io.on("connection") callback > socket.on("connectUser") callback

```javascript
/* Review: Rename to fetchUserById */
async function fetchUser(userID) {
  const query =
    "SELECT user.*, CONCAT(user.firstName, ' ', user.lastName) as userName FROM hm_user user WHERE user.id = ?";
  const queryParams = [userID];
  var result = await executeQuery(query, queryParams);
  return result.length !== 0 ? result[0] : undefined;
}

async function insertUserText(fromUserId, toUserId, text, date) {
  const query =
    "INSERT INTO hm_chat (fromUserId, toUserId, text, createdDate, msgStatus) VALUES (?, ?, ?, ?, ?)";
  const queryParams = [fromUserId, toUserId, text, date, 1];

  /* Review: Remove affectedRows since it is not used any where.*/
  var { affectedRows, insertId } = await executeQuery(query, queryParams);
  return insertId;
}

/* Review: Rename to fetchUsersById */
async function fetchUserTexts(userID) {
  const query = `SELECT
          chat.*,
          CONCAT(toUser.firstName, ' ', toUser.lastName) as toUserName,
          CONCAT(fromUser.firstName, ' ', fromUser.lastName) as fromUserName
    FROM  hm_chat chat
```

File: Chat.js

src > components > chat > Chat.js > Chat

```javascript
export default function Chat(props: Props) {
  /*
    Review: Remove the hardcode image url and place in the common file.
  */
  const pictureUrl = "logo512.png";

  const { showChat, chatClosed, selectedUserId } = props;

  const socket = useRef();
  const textControl = useRef();
  const [texts, setTexts] = useState([]);
  const [arrivalMessage, setArrivalMessage] = useState(null);
  const [selectedChat, setSelectedChat] = useState(
    getDefaultSelectedChat(selectedUserId, texts)
  );
  const currentUser = useSelector(getCurrentUser);

  useEffect(() => {
    if (arrivalMessage) {
      const { userID, userName, id, date, text, inbox } = arrivalMessage;

      let recipientIndex = texts.findIndex((text) => text.userID == userID);

      if (recipientIndex === -1) {
        let newItem = {
```

Developer	Nisha Devi
Reviewer	Rohat Sagar Urif Sonu

File: postController.js

JS postController.js > ...

```javascript
let database = require("../database");
const { validationResult } = require("express-validator");
const util = require("util");

const executeQuery = util.promisify(database.query).bind(database);

module.exports = {
  createPost: async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    // ** COMMENT: View can follow the same naming convention as rest in the code e.g. camelCase
    let {
      Description,
      Status,
      Language,
      SubjectName,
      RatePerHour,
      ExperinceYears,
      AvailableTime,
      UserId,
    } = req.body;

    // ** COMMENT: Can we move this to a util function?
    var date = new Date().toISOString().split("T")[0];
    var isActive = true;

    try {
      let result = await executeQuery(
        "SELECT * FROM hm_tutor_profile T WHERE T.userId = ?;",
        [UserId]
```

JS postController.js > <unknown> > updatePost

```javascript
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    let {
      Id,
      Description,
      TutorProfileId,
      Status,
      Language,
      SubjectName,
      RatePerHour,
      ExperinceYears,
      AvailableTime,
    } = req.body;

    // ** COMMENT: Can we move this to a util function?
    var date = new Date().toISOString().split("T")[0];
    database.query(
      "UPDATE hm_post SET description=?, tutorProfileId=?, status=?, `language`=?, subjectName=?, ratePerHour=?, modifiedDateTime=?, experienceYe
      [
        Description,
        TutorProfileId,
        Status,
        Language,
        SubjectName,
        RatePerHour,
        date,
        ExperinceYears,
        AvailableTime,
        Id,
      ],
      (err, result) => {
```

JS postController.js > <unknown> > updatePost

```javascript
      ],
      (err, result) => {
        if (err) res.status(400).send(`Response Error: ${err}`);
        else res.status(204).json({ message: "Post Details Updated" });
      }
    );
  },

  getPost: async (req, res) => {
    let id = req.params.id;
    database.query(
      "SELECT id, description, tutorProfileId, status, `language`, subjectName, ratePerHour, createdDateTime, modifiedDateTime, experienceYears,
      [id],
      (err, result) => {
        if (err) res.status(400).send(`Response Error: ${err}`);
        else res.status(200).json(result);
      }
    );
  },

  searchPost: async (req, res) => {

    // ** COMMENT: We can move join query logic to a method.
    let joinQuery = "";
    if (req.query.TutorProfileId !== undefined) {
      joinQuery += `tutorProfileId = ${database.escape(
        req.query.TutorProfileId
      )}`;
    }

    if (req.query.Status !== undefined) {
      if (joinQuery != "") joinQuery += " and ";
```

JS postController.js > <unknown> > searchPost

```javascript
      joinQuery += `status = ${database.escape(req.query.Status)}`;
    }

    if (req.query.RatePerHour !== undefined) {
      if (joinQuery != "") joinQuery += " and ";

      // ** COMMENT: We should add check for greater than or equal
      joinQuery += `ratePerHour = ${database.escape(req.query.RatePerHour)}`;
    }

    if (req.query.SubjectName !== undefined) {
      if (joinQuery != "") joinQuery += " and ";

      // ** COMMENT: Is this a case-sensitive?
      joinQuery += `MATCH(subjectName) AGAINST (${database.escape(
        req.query.SubjectName
      )})`;
    }

    let dbQuery =
      "SELECT hm_post.id, hm_post.description, hm_post.tutorProfileId, hm_post.status, hm_post.language, hm_post.subjectName, hm_post.ratePerHour
      " INNER JOIN hm_tutor_profile ON (hm_tutor_profile.id = hm_post.tutorProfileId)" +
      " INNER JOIN hm_user ON (hm_user.id = hm_tutor_profile.userId)";
    if (joinQuery !== "") dbQuery += ` where ${joinQuery}`;

    database.query(dbQuery, (err, result) => {
      // ** COMMENT: In case of error I think we should return some error.
      if (err) console.log(err);
      else res.json(result);
    });
  },
};
```

Developer	Hasib Iqbal
Reviewer	Chowdhury Amlan Barua
File Name	ManageTutorsProfile.js tutor.js

File Name: ManageTutorsProfile.js

```
import React from "react";
import { useSelector } from "react-redux";
import { ListGroup } from "react-bootstrap";
import TutorProfileItem from "./TutorProfileItem";
import { getTutorsProfileList } from
"../../../../core/selectors/manageTutorsProfile";
import Page from "../../../../components/page/Page";
import FilterBar from "./filterBar/FilterBar";

// Destructuring the props might be a good idea. You can do this with the
reference below:
// https://medium.com/@lcriswell/destructuring-props-in-react-b1c295005ce0

function ManageTutorsProfile(props) {
 var data = useSelector(getTutorsProfileList);

 if (data === undefined) {
   return <div></div>;
 }

 return (
   <Page>
     <FilterBar />
     <br />
     <ListGroup>
       {data?.map((item, i) => {
         return <TutorProfileItem key={i} item={item} />;
       })}
     </ListGroup>
     <br />
   </Page>
```

```
 );
}

export default ManageTutorsProfile;
```

File Name: tutor.js

```
export function* getTutorList(action: Object): Saga<void> {
 const { filters } = action.payload;

 var url = allTutorListApi;

 if (filters.fName) {
   url += `&FirstName=${filters.fName}`;
 }
 if (filters.lName) {
   url += `&LastName=${filters.lName}`;
 }
 if (filters.email) {
   url += `&Email=${filters.email}`;
 }


 const apiOptions: ApiOptions = {
   url: url,
   method: "GET",
   useJwtSecret: false,
 };

 const apiResponse: ApiResponse = yield call(executeApiCall, apiOptions);

 const { isSuccessful, response = {} } = apiResponse;

 if (isSuccessful) {
   var data = response;
   yield put(getTutorListSuccess({ data }));
 } else {
   var msg = "Failed to load data from API"; //A more descriptive error
message might be constructed
```

```
    yield put(getTutorListFailed({ msg }));
  }
}
```

Developer	Mohammad Rakibul Hasan
Reviewer	Hasib Iqbal
File Name	AddQualification.js

```
function AddQualification(props) {
  const dispatch = useDispatch();

  const subjectRef = useRef(null);
  const qualificationRef = useRef(null);
  const gradeRef = useRef(null);
  const descriptionRef = useRef(null);

  const user = useSelector(getCurrentUser);
  console.log("userid" + user );

  // Review Comment:
  // 1.Follow Javascript naming convention for variable names (Ref>
https://www.w3schools.com/js/js_conventions.asp)
  // 2.Add comments for better code readability
  // 3. Remove unnecessary codes


  //function to save the qualification
  const submitQualification = () => {
    const qualification = {
      SubjectName: subjectRef.current.value,
      Grade: gradeRef.current.value,
      Description: descriptionRef.current.value,
      UserId: user.id
    };
```

```
    console.log(qualification);
    dispatch(saveQualification(qualification));
    //test
    // dispatch(fetchQualificationById(1));
  };

  return (
    <div>
    <Page></Page>
    <div className="qualification-page">
      <div className="qualification-content">
        <h1>Add Qualification</h1>
        <Form>
          <br />
          <Form.Control type="text" ref={subjectRef} placeholder="Subject"
/>
          <br />
          {/* <Form.Control type="text" ref={qualificationRef}
placeholder="Qualification" />
          <br /> */}
          <Form.Control type="text" ref={gradeRef} placeholder="Grade" />
          <br />
          <Form.Control
            ref={descriptionRef}
            as="textarea"
            rows={3}
            placeholder="Description"
          />
          <Button className="btn btn-success" variant="primary"
onClick={submitQualification} type="submit">
            Save
          </Button>
        </Form>
      </div>
    </div>
    </div>
  );
}
```

Developer	Mohammad Salman Haydar
Reviewer	Talha Jahangir Khan
File Name	UploadController.js

controller > JS uploadController.js > upload

```javascript
const uploadFile = require("../middleware/upload");
const database = require("../database");
const util = require("util");
require("dotenv").config();

const executeQuery = util.promisify(database.query).bind(database);

const upload = async (req, res) => {
  try {

    await uploadFile(req, res);

    if (req.file == undefined) {
      return res.status(400).send({ message: "Please upload a file!" });
    }

    var result = await executeQuery('SELECT id FROM hm_tutor_profile WHERE userId = ?', [req.userid]
    var tutorProfileId = result[0].id;

    if(req.file.mimetype === "application/pdf") {

        database.execute("SELECT * FROM `helpmelearn`.`hm_file` WHERE `tutorProfileId`= ?",
        [tutorProfileId],
        (err, result) => {
            if(err) {
                console.log(err);
                res.status(500).send({message:"Something went wrong"});
            }
```

```javascript
            }
            else if(result.length >= 1) {
              database.execute("DELETE FROM `helpmelearn`.`hm_file` WHERE (`tutorProfileId` = ?)",
              [tutorProfileId],(err, result)=> {
                    if(err) {
                      console.log(err);
                      res.status(500).send({message:"Something went wrong"});
                  }
                  else {
                    database.execute("INSERT INTO `helpmelearn`.`hm_file` ( `tutorProfileId`, `fileN
                    [tutorProfileId,
                    req.file.originalname,
                    0,
                    "pdf",
                    "resources/static/"+req.file.originalname],
                    (err, result) => {

                      if (err){
                        console.log(err);
                        res.status(500).send({message: "Somethid went wrong during inserting into DB
                      }

                      res.status(200).send({
                        message: "Uploaded the file successfully: " + req.file.originalname,
                      });
                    });
                  }
              });
            }
```

```
            }
            else {

              database.execute("INSERT INTO `helpmelearn`.`hm_file` ( `tutorProfileId`, `fileName`,
              [tutorProfileId,
              req.file.originalname,
              0,
              "pdf",
              "resources/static/"+req.file.originalname],
              (err, result) => {

                if (err){
                  console.log(err);
                  res.status(500).send({message: "Somethid went wrong during inserting into DB"});
                }

                res.status(200).send({
                  message: "Uploaded the file successfully: " + req.file.originalname,
                });
              });
            }

          });

      }
```

```
  }
  else if(req.file.mimetype === "image/jpg" || req.file.mimetype === "image/jpeg" || req.file.mimetype === "image/png") {

      var today = new Date();
      var date = today.getFullYear()+'-'+(today.getMonth()+1)+'-'+today.getDate();
      var time = today.getHours() + ":" + today.getMinutes() + ":" + today.getSeconds();
      var dateTime = date+' '+time;
      console.log(req.userid);
      database.execute("SELECT * FROM `helpmelearn`.`hm_image` WHERE `userId`= ?",
      [req.userid],
      (err, result) => {
          if(err) {
              console.log(err);
              res.status(500).send({message:"Something went wrong"});
          }
          else if(result.length >= 1) {
            database.execute("DELETE FROM `helpmelearn`.`hm_image` WHERE (`userId` = ?)",
            [req.userid],(err, result)=> {
                  if(err) {
                    console.log(err);
                    res.status(500).send({message:"Something went wrong"});
                }
                else {
                  database.execute("INSERT INTO `helpmelearn`.`hm_image` ( `imagePath`, `date`, `userId`, `createdDateTime`,
                  ["resources/static/"+req.file.originalname,
                  dateTime,
                  req.userid,
                  dateTime,
                  dateTime],
```

```
                (err, result) => {
                    if (err){ console.log(err);
                        res.status(500).send({message: "Somethid went wrong during inserting into DB"});
                    }
                    res.status(200).send({
                        message: "Uploaded the image successfully: " + req.file.originalname,
                        });
                });
                }
            });
        }
        else {

            database.execute("INSERT INTO `helpmelearn`.`hm_image` ( `imagePath`, `date`, `userId`, `createdDateTime`, `modif
            ["resources/static/"+req.file.originalname,
            dateTime,
            req.userid,
            dateTime,
            dateTime],
            (err, result) => {
                if (err){ console.log(err);
                    res.status(500).send({message: "Somethid went wrong during inserting into DB"});
                }
                res.status(200).send({
                    message: "Uploaded the image successfully: " + req.file.originalname,
                    });
            });
        }
    });
}
```

```
        }
        else {

            database.execute("INSERT INTO `helpmelearn`.`hm_image` ( `imagePath`, `date`, `userId`, `createdDateTime`, `modif
            ["resources/static/"+req.file.originalname,
            dateTime,
            req.userid,
            dateTime,
            dateTime],
            (err, result) => {
                if (err){ console.log(err);
                    res.status(500).send({message: "Somethid went wrong during inserting into DB"});
                }
                res.status(200).send({
                    message: "Uploaded the image successfully: " + req.file.originalname,
                    });
            });
        }
    });
}

catch (err) {
res.status(500).send({
  message: `Could not upload the file: ${req.file?.originalname}. ${err}`,
});



ıle.exports = {
 "upload" : upload,
```

Developer	Chowdhury Amlan Barua
Reviewer	Hasib Iqbal
File Name	TutorprofilController.js

server > controller > JS TutorProfileController.js

```
4   require("dotenv").config();
5   const util = require("util");
6
7   const executeQuery = util.promisify(database.query).bind(database);
8
9   module.exports = {
10
11    // Review: In case of error handling,  I think, we should use try catch functionality
12    getTutorAbouInfoById: async (req, res) => {
13      let id = req.params.id;
14      let query = `SELECT firstName, lastName, about, age, picPath FROM hm_tutor_profile A, hm_user B WHERE A.userId = B.id AND userI
15
16      database.query(query, [id], (err, result) => {
17        if (err) console.log(err);
18        else res.json(result);
19      });
20    },
21
22    // Review: 1. In case of error handling,  I think, we should use try catch functional"ity
23    //        2. Remove console.log as it is not doing anything.
24    getTutorOfferedCoursesById: async (req, res) => {
25      let id = req.params.id;
26      let query = `SELECT subjectName, ratePerHour FROM hm_post A inner join hm_tutor_profile B on
27      (A.tutorProfileId = B.id and B.userId = ?);`;
28      console.log(query);
29      database.query(query, [id], (err, result) => {
30        if (err) console.log(err);
31        else res.json(result);
32      });
33    },
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell

server > controller > JS TutorProfileController.js

```
33
34    // Review: I think, using handling error with try catch would be better and in case of error, it should return error
35    getTutorQualificationById: async (req, res) => {
36      let id = req.params.id;
37      let query = `SELECT A.id, A.subjectName, A.description, A.grade FROM hm_qualification A
38       inner join hm_tutor_profile B on (A.tutorProfileId = B.id and B.userId = ?);`;
39
40      database.query(query, [id], (err, result) => {
41        if (err) console.log(err);
42        else res.json(result);
43      });
44    },
45
46    // Review: I think, It's preferable to put all code that can cause errors inside try block for efficient error handling.
47            // I think, for inner joing query proper using meaningful variable name be better for code readability.
48    getReviewsById: async (req, res) => {
49      let id = req.params.id;
50      try {
51        let result = await executeQuery(
52          `SELECT A.id, A.text, A.rating, A.createdDateTime, A.modifiedDateTime, U.firstName, U.lastName, A.userId FROM hm_review A
53        inner join hm_user U on (A.userId = U.id)
54        inner join hm_tutor_profile T on (A.tutorProfileId = T.id and T.userId = ?);`,
55          [id]
56        );
57
58        res.status(200).json(result);
59      } catch (error) {
60        res.status(500).json({ message: error });
61      }
62    },
```

server > controller > TutorProfileController.js

```javascript
  // Review: 1. In case of error handling,  I think, we should use try catch functionality
            // 2. For Picture path, I think, instead of typing image path, using global variable in common file
            //    would be better for minimizing typing error issue.
  saveTutorInfo: async (req, res) => {
    await uploadFile(req, res);
    if (req.file == undefined) {
      return res.status(400).send({ message: "Please upload a Image!" });
    }

    let { UserId, About, Age } = req.body;
    let PicturePath = "public/images/" + req.file.originalname;
    database.query(
      "UPDATE hm_tutor_profile SET about = ?, age = ?, rating = 0, picPath = ?, status = 100 WHERE userId = ?",
      [About, Age, PicturePath, UserId],
      (err) => {
        if (err) res.status(400).send(`Response Error: ${err}`);
        else res.status(200).json({ message: "Tutor profile updated" });
      }
    );
  },

   // Review: 1. In case of error handling,  I think, we should use try catch functionality
  updateTutorInfo: async (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    let { UserId, Status } = req.body;
    database.query(
      `UPDATE hm_tutor_profile SET status = ? WHERE userId = ?`,
      [Status, UserId],
      (err) => {
        if (err) {
          res.status(500).json({ message: error });
        } else {
          res.json({ message: "Tutor Profile Updated" });
```

Developer	Chowdhury Amlan Barua
Reviewer	Mohmmad Rakibul Hasan
File Name	ReviewList.js

client > src > pages > tutorProfile > reviewList > JS ReviewList.js

```javascript
import React, { useState, useEffect, useRef } from "react";
import { useSelector, useDispatch } from "react-redux";
import { useParams } from "react-router-dom";
import moment from "moment";
import { getTutorReviewDataById } from "../../../core/selectors/tutor";
import { getTutorReviewById } from "../../../core/actionCreators/tutor";
import { setTutorReview } from "../../../core/actionCreators/tutor";
import { saveReview } from "../../../core/actionCreators/tutor";
import { ListGroup, Row, Col, Button, Form } from "react-bootstrap";
import Rate from "rc-rate";        Rohat Sagar, 2 months ago • redesign tutorProfile UI, fixed login and registr…
import "rc-rate/assets/index.css";
import { getCurrentUser, getUserType } from "../../../core/selectors/user";

// Destructuring props would be a good idea instead of using whole props
export default function ReviewList(props) {
  const dispatch = useDispatch();

  let starCountRef = useRef(null);
  const textReviewRef = useRef(null);
  const user = useSelector(getCurrentUser);
  const userType = useSelector(getUserType);

  // As we are using react router so, using useparams hook to get tutorId would be cleaner to do that.

  let { tutorId } = useParams();
  if (props.tutorId !== undefined && props.tutorId != "") {
    tutorId = props.tutorId;
  }
  const tutorReviewData = useSelector(getTutorReviewDataById);
  const [tutorReviews, setTutorReview] = useState([]);

  useEffect(() => {
    dispatch(getTutorReviewById(tutorId));
  }, []);
  useEffect(() => {
    setTutorReview(tutorReviewData);
  }, [tutorReviewData]);
```

client > src > pages > tutorProfile > reviewList > JS ReviewList.js

```javascript
  const submitReview = () => {
    let review = {
      Rating: starCountRef.current.state.value,
      Text: textReviewRef.current.value,
      UserId: user.id,
      TutorProfileId: Number(tutorId),
    };
    dispatch(saveReview(review));
  };

  const renderReview = () => {
    if (userType !== "student") return null;

    return (
      <div>
        <Row>
          <span>YOUR REVIEW</span>
          <Rate
          // May be using global variable for default value instead of hard coding deafault value?      You, seconds
            defaultValue={2.5}
            ref={starCountRef}
            allowHalf
            allowClear={false}
          />
          <Col sm={11}>
            <Form.Control size="md" ref={textReviewRef} type="text" />
          </Col>
          <Col sm={1}>
            <Button
              className="float-end"
              variant="primary"
              size="md"
              onClick={submitReview}
              type="submit"
            >
              Submit
            </Button>
```

```
83    const renderReviews = () => {
84      // I think, when we return a Null component we still have a full lifecycle that will trigger depending on what we do on their parent
85      // component. I think, more correct way to do is to do the conditionals on the parent component to avoid even call that child component
86      if ( tutorReviews === undefined || tutorReviews.length === undefined || tutorReviews.length === 0) { return null; }        You, seconds ago •
87      return (
88        <div>
89          <span>REVIEWS</span>
90          {/* May be move that to a css class? */}
91          <ListGroup style={{ padding: "1.0rem 0 0 0" }}>
92            {tutorReviewData?.map((item, i) => {
93              return (
94                <ListGroup.Item
95                  key={i}
96                  className="d-flex justify-content-between align-items-start"
97                >
98                  <div className="me-auto">
99                    <div className="fw-bold">{`${item.firstName} ${item.lastName}`}</div>
100                   <div>
101                     <Rate defaultValue={item.rating} disabled />
102                     <span className="text-muted">{item.modifiedDateTime}</span>
103                   </div>
104                   <div className="fw-light">{item.text}</div>
105                 </div>
106               </ListGroup.Item>
107             );
108           })}
109         </ListGroup>
110       </div>
111     );
112   };
113   return (
114     <div>
115       {renderReview()}
116       {renderReviews()}
117     </div>
118   );
119 }
```

Self Check on best practices for security

List of major assets that we should protect

- Passwords
- Admin routes
- Private routes

List of major threats for each asset above

- **Passwords:** should be encrypted, otherwise if the database is hacked and the passwords are stored as a plain text, then all the accounts will be exposed.
- **Admin routes:** admin functionalities are critical and only authorized users who have admin privilege can access these routes, otherwise, any user can manipulate the posts and site users improperly.
- **Private routes:** only authenticated users should access these routes like the chatting and adding posts routes.

For each asset, how we may protect it

- **Passwords:** using encryption, so all passwords are hashed and saved in the database.
- **Admin routes:** verifying the token sent in the request header and checking the user role before allowing him to access the API using the "checkAdmin" middleware.
- **Private routes:** verifying the token sent in the request header and checking if it's valid before allowing him to access the API using the "checkAuth" middleware.
- On the client side the admin and private routes are not accessed by not logged in users.

Confirm that you encrypt PW in the DB

We are saving the passwords hashed and not as plain text in the database using the Blowfish Cipher which is one way hashing and cannot be converted to the plain text password.

Confirm Input data validation

- **Valid Email address:** a checking function is used in the signup to verify that the email address has a valid format and is related to Fulda/San Francisco university.
- **Strong Rules for passwords:** a checking function is used in the signup to enforce the user to choose a strong password that complies with the rules of the website (8 min length, one small, one capital one digit and one special char).
- **Limit the search field for up to 40 characters max:** limiting the size of the input field to 40 characters max, and preventing the user to exceed this number.

Self-check: Adherence to original Non-functional specs

List of non functional requirements	Done	On Track
Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in Milestone 0. Application delivery shall be from chosen cloud server	✔	
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	✔	
All or selected application functions must render well on mobile devices	✔	
Data shall be stored in the database on the team's deployment cloud server	✔	
No more than 50 concurrent users shall be accessing the application at any time	✔	
Privacy of users shall be protected and all privacy policies will be appropriately communicated to the users.	✔	
The language used shall be English (no localization needed)	✔	
Application shall be very easy to use and intuitive	✔	
Application should follow established architecture patterns		✔
Application code and its repository shall be easy to inspect and maintain	✔	
No email clients shall be allowed.	✔	
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	✔	
Site security: basic best practices shall be applied (as covered in the class) for main data items	✔	
Application shall be media rich (images, video etc.). Media formats shall be standard as used in the market today		✔
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	✔	
For code development and management, as well as documentation like formal milestones required in the class, each team shall use their own GitHub to be set-up by class instructors and started by each team during Milestone 0	✔	
The application UI (WWW and mobile) shall prominently display the following exact text on all pages "Fulda University of Applied Sciences Software Engineering Project, Fall 2021 For Demonstration Only" at the top of the WWW page.	✔	