
Teknikattan scoring system

Albin Henriksson, Sebastian Karlsson, Victor Löfgren, Björn Mod

Jun 08, 2021

CONTENTS

1	Introduction	3
2	User manual	5
3	System overview	15
4	Installation	21
5	Development	25
6	Testing	29
7	Documentation	31
8	Contact	33
9	License	35

This project was developed during the period January-May 2021 as part of the course [TDDD96 Kandidatprojekt i programvaruutveckling](#) at Linköping University. It was developed for [Teknikåttan](#).

INTRODUCTION

This is a short introduction to both the project as a whole and our system.

1.1 Introduction to the project

This is a short introduction to the project. There are several links to other relevant things to read before choosing this project.

1.1.1 Before choosing this project

There are a lot of things this system needs to do. To get a complete description, see the [original repository](#) from Teknikåttan. There you will see exactly what is expected of the system (click on each picture to see a video that will give a more in-depth explanation). You may also want to look at the [description of the project](#), if you have not already done so. There is a lot to read (and watch) on these two links, but in doing so you will get a complete picture of the requirements. Make sure you understand what this project entails before continuing with it, it is not as “simple” as it might first seem.

1.1.2 Our perspective

This was a fun project. In contrast to some other previous projects the purpose of this one, what its requirements are and why its useful, is clear. It is really fun developing a product you know (if it turns out well) many people will appreciate, use, and see.

But on the other hand the project is large. There was a group that worked on this project before us. We could have continued their project when we began, but we decided not to. This was in part due to it not really working and in part due to lack of documentation. We hope to have learned from that mistake. That is why we have made proper documentation (the one you are reading right now!) and a decent, working foundation of the system. We have also made an effort to document the code as much as possible. We hope you continue on our efforts if you choose this project.

1.1.3 Contact us

If you have any questions about the project, our system or anything, feel free to [contact](#) any of us.

1.2 Introduction to our system

This system allows a user to create, edit and host competitions. Below it is in short described what the system allows you to do. If you want a more exact description (with pictures!), see the [user manual](#)

1.2.1 Login

After logging in you will be able to see all competitions and edit them. If you're an admin you will also be able to see all users and edit them. You will also be able to connect to an active competition from the same screen you used to login.

1.2.2 Editor

The editor allows you to edit competitions. You can add, remove and reorder slides. You can add, delete and edit:

- teams
- text and image components
- questions
- question types
- correcting instructions
- background image.

1.2.3 Active competitions

You can also start a competition. This will let other people join it with codes that can be seen either before or after starting a presentation. Then when you switch slides, start the timer or show the current score, it will also happen for every other person connected to the same competition.

Depending on which code someone uses to join an active competition they will see different things, which we call different *views*. The *team view* will allow the user to answer the questions. The *judge view* will allow the user to see correct answers and give a score to the questions answered by a team. The *audience view* will show the current slide.

USER MANUAL

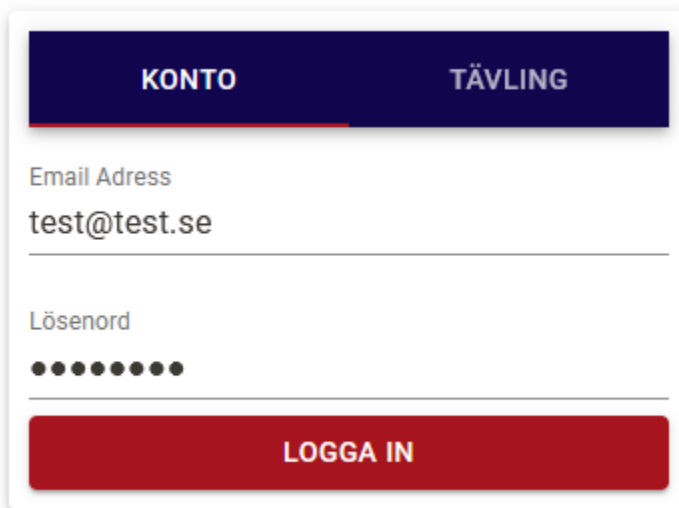
The user manual will describe how to login, navigate the admin page, create and edit a competition and host and participate in a presentation.

2.1 Login

The login page will let you either login as a user or join a competition with a code.

2.1.1 User

The first page you will be presented with when accessing the site is the login page. From here you can login with your account by typing your email and password in their respective fields and pressing the “Logga in” button.

A login form mockup with a dark blue header bar containing two tabs: 'KONTO' (selected) and 'TÄVLING'. Below the header, there are two input fields. The first is labeled 'Email Adress' and contains the text 'test@test.se'. The second is labeled 'Lösenord' and contains ten black dots representing a password. At the bottom of the form is a red button with the text 'LOGGA IN' in white capital letters.

KONTO TÄVLING

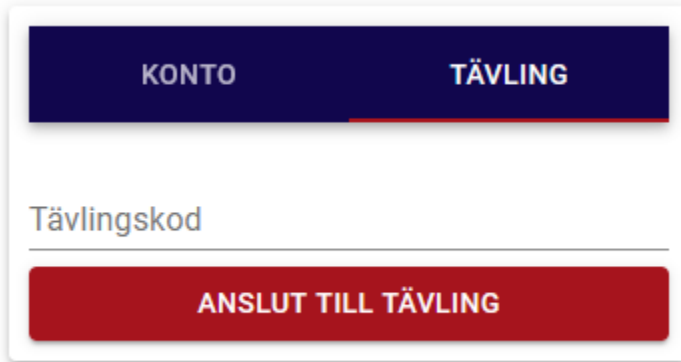
Email Adress
test@test.se

Lösenord
●●●●●●●●●●

LOGGA IN

2.1.2 Competition code

You can also choose the “Tävling” tab. Here you can enter your six character long code and by pressing the “Anslut till tävling” button you will be able to join a competition.

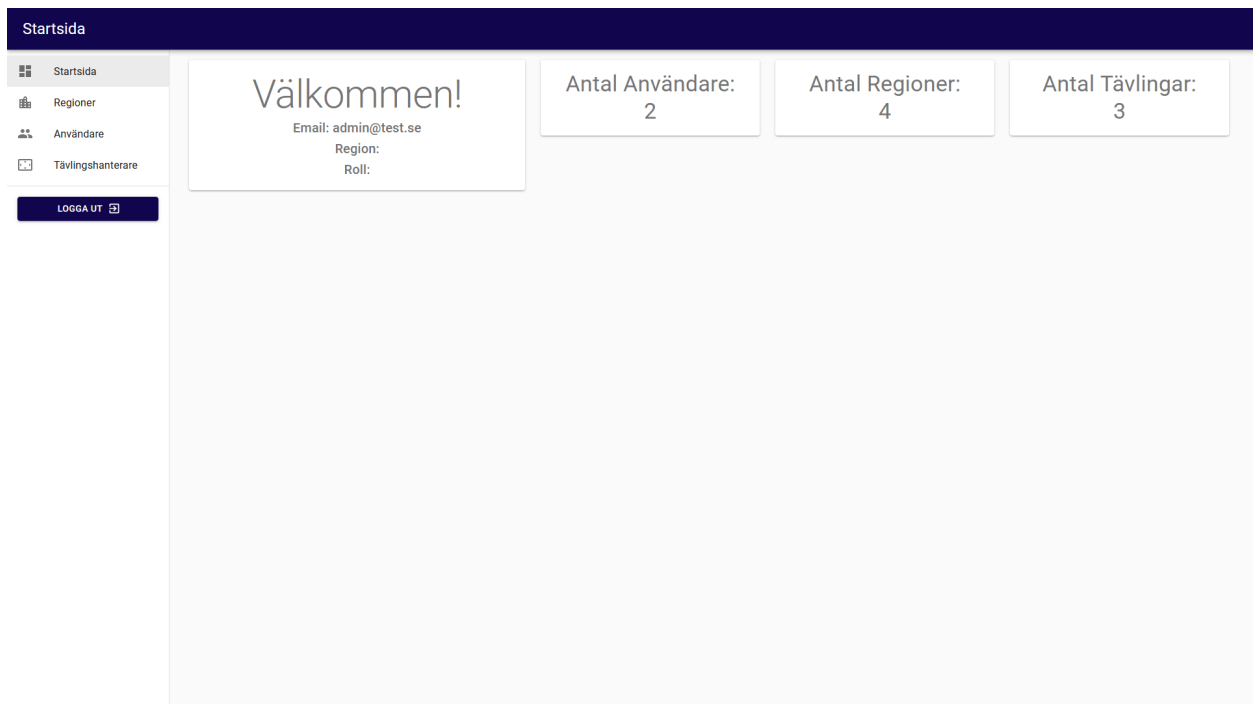


The screenshot shows a web interface with two tabs at the top: "KONTO" and "TÄVLING". The "TÄVLING" tab is selected. Below the tabs is a text input field labeled "Tävlingskod". At the bottom of the form is a red button with the text "ANSLUT TILL TÄVLING".

These codes can be accessed from [Admin](#).

2.2 Admin

After logging in you will see the admin page. To the left you will see the start page and the competitions tab. If you are an admin you will also see regions and users. In the bottom left you will be able to logout by pressing the “Logga ut” button.



The screenshot shows the admin dashboard. At the top is a dark blue header with the text "Startsida". Below the header is a sidebar on the left with a menu containing "Startsida", "Regioner", "Användare", and "Tävlingshanterare". At the bottom of the sidebar is a "LOGGA UT" button. The main content area has a large "Välkommen!" greeting, followed by user details: "Email: admin@test.se", "Region:", and "Roll:". To the right of the greeting are three boxes showing statistics: "Antal Användare: 2", "Antal Regioner: 4", and "Antal Tävlingar: 3".

2.2.1 Regions

The regions tab will show all regions. To create a new region, enter its name at the top and then click the “+” button.

Regioner

Startsida

Regioner

Användare

Tävlingshanterare

LOGGA UT

Region

Regioner

Linköping	...
Stockholm	...
Norrköping	...
Örskelljunga	...

2.2.2 Users

The users tab will allow you to see all users, their name, region and role. You will also be able to create new users by clicking the “Ny användare” button. By clicking the three dots “...” you will be able to edit or delete that user. You will also be able to search for and filter users by their region or role.

Användare

Startsida

Regioner

Användare

Tävlingshanterare

LOGGA UT

Sök

Region
Alla

Roller
Alla

NY ANVÄNDARE

Email	Namn	Region	Roll	
admin@test.se		Linköping	Admin	...
test@test.se		Linköping	Editor	...

1-2 of 2 < >

2.2.3 Competitions

The competitions tab will allow you to see all competitions, their name, region and year. You will also be able to create a new competition by clicking the “Ny tävling” button or edit existing ones by clicking on their name. By clicking on the three dots “...” you will be able to start, show the codes for, copy or delete that competition.

Tävlingshanterare

Startsida

Regioner

Användare

Tävlingshanterare

LOGGA UT

Sök

Region
Alla

År
År

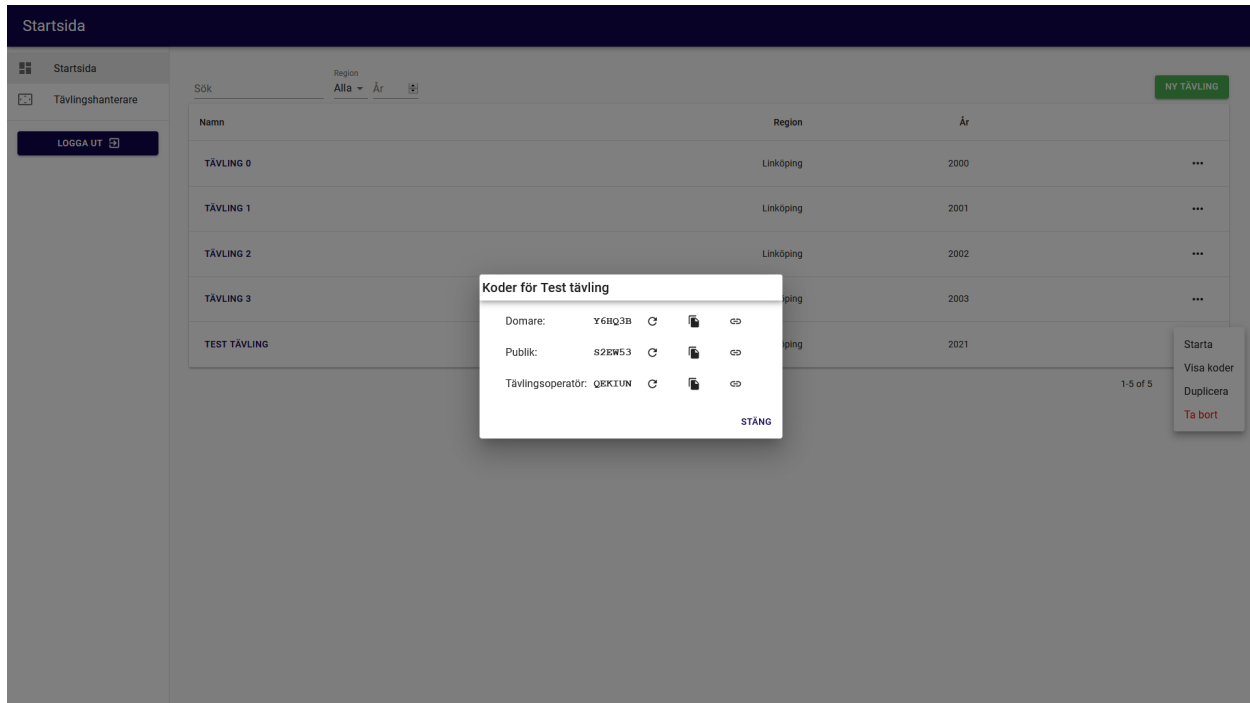
NY TÄVLING

Namn	Region	År	
TÄVLING 1	Linköping	2001	...
TÄVLING 2	Linköping	2002	...

1-2 of 2 < >

Competition codes

By pressing the three dots “...” for a competition and then pressing “Visa koder”, all the codes for that competition will be shown. Here you will see what view each code is associated with and what the code is. You will also be able to generate a new code, copy the code or copy a link to the code that will let others join, or even host, a competition directly.



2.3 Editor

The *competition manager* will list all competitions. After clicking on a competition name you will enter the editor and will be able to edit it. The Teknikåttan logo in the top left corner will take you back to the Admin page and right under that all slides are shown. A newly created competition will have one empty default slide. Switch to a different slide by clicking on it. In the bottom left corner you will be able to add a new slide using the “Ny sida” button. Delete or copy a slide simply by right clicking on it and choosing the appropriate option. In the top right corner you will be able to change which view you see and edit. By right clicking on a component you will be able to delete it or copy it to the same or a different view.

The screenshot shows the 'Test tävling' (Test competition) settings page in the Teknikattan scoring system. The interface is divided into three main sections: a left sidebar, a central content area, and a right sidebar.

- Left sidebar:** Contains a 'Sida 1' (Slide 1) tab and a 'Ny sida +' (New slide +) button at the bottom.
- Central content area:** A large, empty white space for editing the competition details.
- Right sidebar:** Contains the following elements:
 - Tävlingens namn:** A text input field with the value 'Test tävling'.
 - Region:** A dropdown menu with 'LINKÖPING' selected.
 - Lag:** A text input field.
 - LÄGG TILL LAG:** A button to add a new team.
 - VÄLJ BAKGRUNDSBILD...:** A button to select a background image.

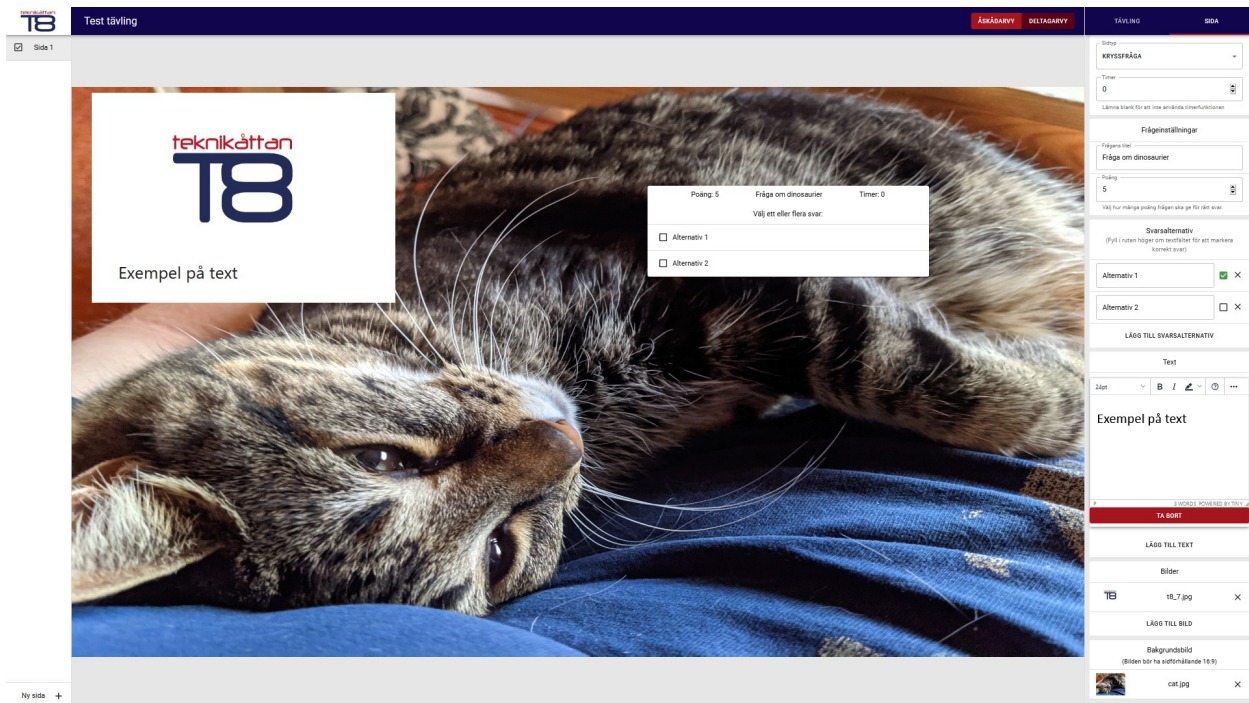
At the top of the interface, there is a dark blue header bar with the text 'Test tävling' and two red buttons: 'ÅSKÅDARVY' and 'DELTAGARVY'. Below the header bar, there are two tabs: 'TÄVLING' (active) and 'SIDA'.

2.3.1 Competition settings

To the right you will see the active tab “Tävling”, which will show and let you edit everything about the entire competition. There you will be able to edit the competition name, add a new team and a background image. The background image for the competition will be used for all slides in the competition.

2.3.2 Slide settings

If you choose the “Sida” tab, you will be able to edit the current slide. In the top right you can change the question type of the current slide. For all question types you will be able to add a timer for how long the teams have to answer that question. Depending on which type you choose, you will have different options below. For this example we will choose multiple choice (“Kryssfråga”). For this question type you will have the option to add a title to the question and how much many points a correct answer yields. For this question type you will also be able to add alternatives, which the teams will be able to choose between during a competition. Below that you will be able to add and remove text and image components as well as a background image. The background image for the competition can be overridden by explicitly setting it on a specific page.



2.4 Presentations

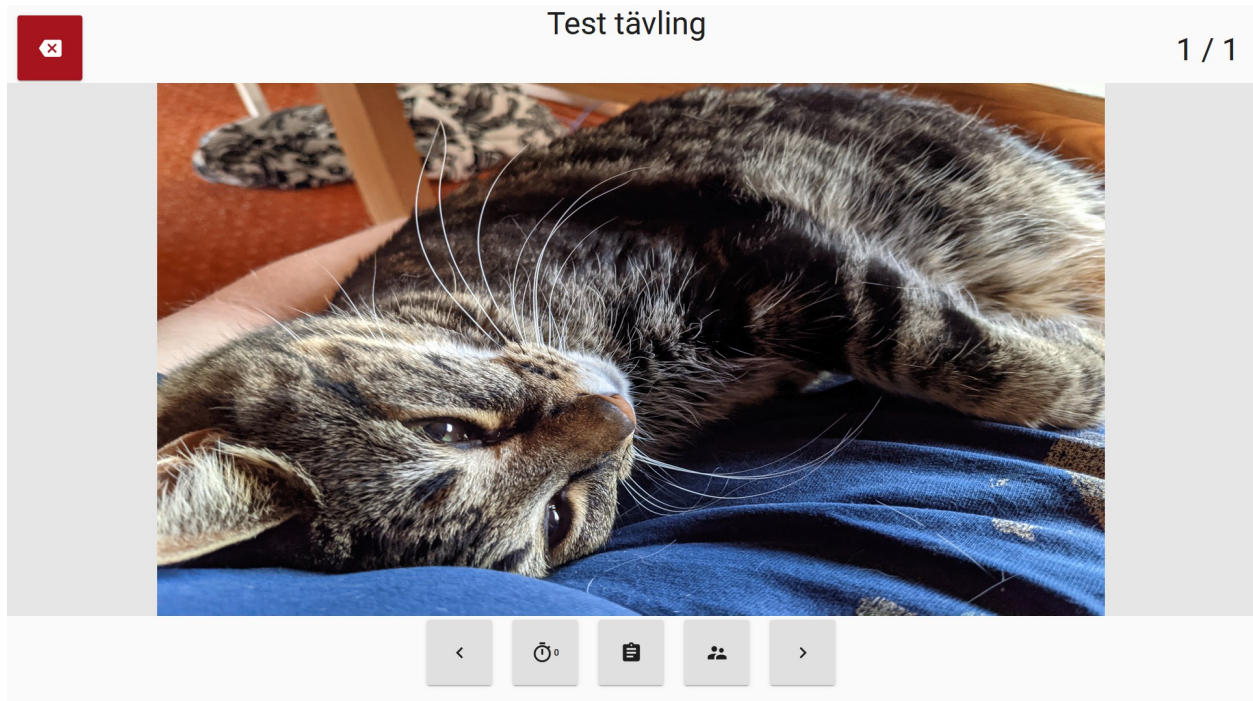
An active (i.e. started) competition is for simplicity's sake called a presentation. There are many different views during a presentation. Below it is described how to start a competition, how to join a presentation, and how the different kinds of views work.

2.4.1 Competition codes

You can join a presentation with codes. This can either be done by pasting the link that can be copied when listing the codes or can be typed by hand in the login page. All the views have different purposes and therefore looks a little bit different from one another.

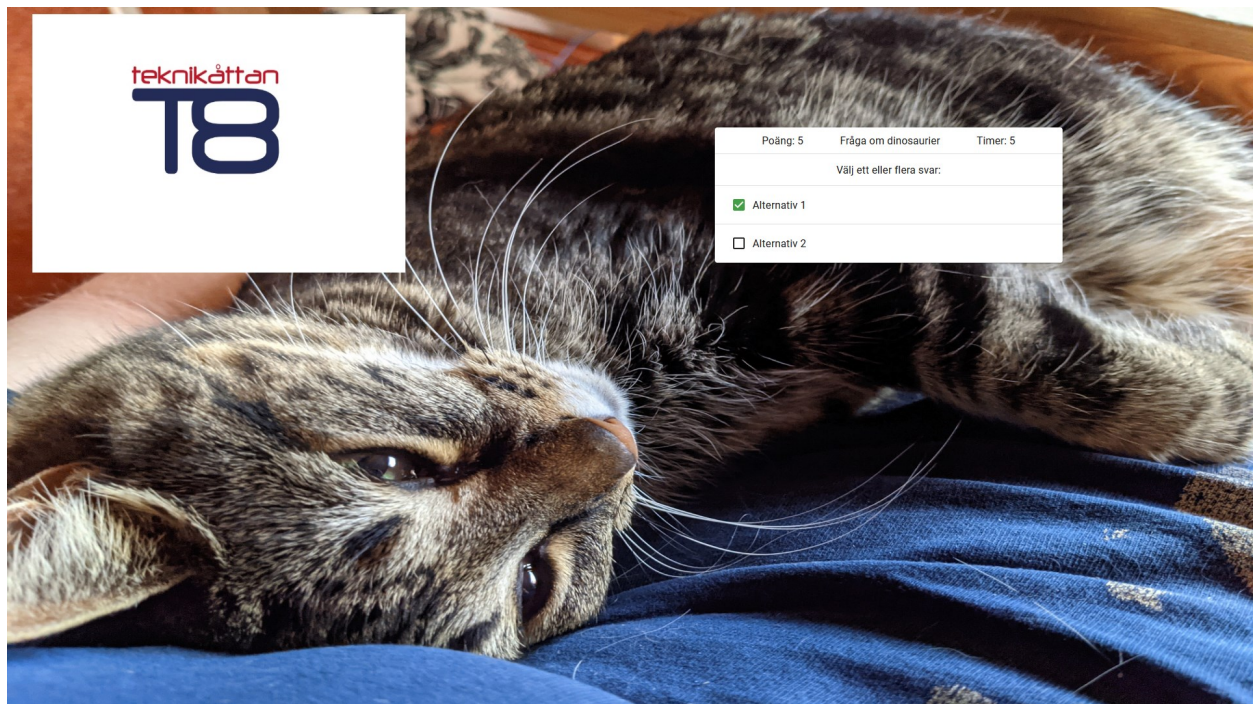
2.4.2 Operator

There are two ways to start a competition. The first way is to navigate to the competition manager, press the three dots "...", and press "Starta". You will then enter the operator view. From there you will be able to go between slides with the "<" and ">" buttons and start the timer, both will be synced between all clients connected to that presentation. You will also be able to view all codes to the competition. You can also show the current score for all teams to the audience.



2.4.3 Team

The team view will be used by teams. It shows the current slide (that the operator has decided) and allows the user to answer questions on the slide, which will be saved.



2.4.4 Audience

The audience view will look like the operator view but without the buttons.



2.4.5 Judge

The judge view will show the same slide as team view. To the left you will be able to move between different slides without affecting the other clients and will be shown on which slide the operator currently is. To the right you will see what the teams have answered on every question, what score each team got on each question, their total score and be able to set the score of a team on any and all questions. In the bottom right you will see instructions for how to grade the current question.

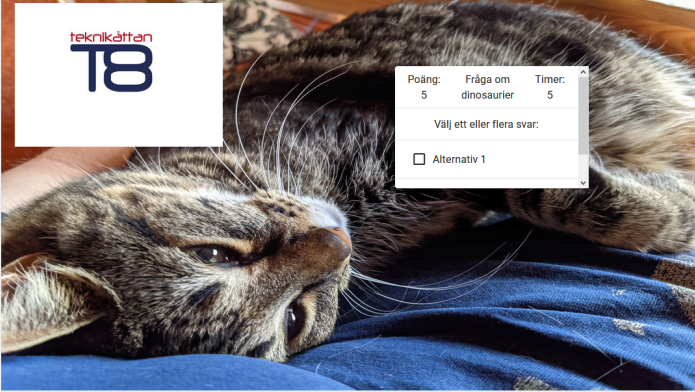
Frågor

Svar

☒ Sida 1

☐ Sida 2

☐ Sida 3



Fråga om dinosaurier (5p)

Albins gymnasieskola deluxe

Poäng: 0

Alla poäng: [0]

Total poäng: 0

Alternativ 1

Testlag 2

Alla poäng: []

Total poäng: 0

Inget svar

Testlag 3

Alla poäng: []

Total poäng: 0

Inget svar

Rättningsinstruktioner

Det finns inga rättningsinstruktioner för denna fråga

SYSTEM OVERVIEW

This is a brief overview of how the entire system works. There is then more detail about the client, the server and the database.

3.1 System overview

The system has a fairly simple design, as depicted in the image below. The terms frontend and client as well as backend and server will be used interchangeably.

First there is the main server which is written in Python using the micro-framework Flask. Then there is a fairly small Node server with only one function, to serve the React frontend pages. Lastly there is the frontend which is written in TypeScript using React and Redux.

3.1.1 Communication

The frontend communicates with the backend in two ways, both of which are authorized on the server. This is to make sure that whoever tries to communicate has the correct level of access.

API

API calls are used for simple functions that the client wants to perform, such as getting, editing, and saving data. These are sent from the client to the backend Node server that will forward the request to the main Python server. The request will then be handled there and the response will be sent back. The Node server will then send them back to the client.

Sockets

The client can also communicate directly with the server via sockets. These are suited for fast real time communication. Thus they are used during an active presentation to sync things between different views such as current slide and timer.

3.2 Client overview

The client is the main part of the system. It is divided into 4 pages: login, admin, presentation editor and active competitions (presentations). The presentations is also further divided into four different views: operator view, audience view, team view and judge view.

3.2.1 Competitions and Presentations

In this project competitions are often referred to when meaning un-active competitions while presentations are referred to when meaning active competitions involving multiple users and sockets connecting them.

3.2.2 File structure

All of the source code for the pages in the system is stored in the `client/src/pages/` folder. For each of the different parts there is a corresponding file that ends in `Page`, for example `JudgeViewPage.tsx` or `LoginPage.tsx`. This is the main file for that page. All of these pages also has their own and shared components, in the folder relative to the page `./components/`. Every React component should also have their responding test file.

3.2.3 Routes

All pages have their own route which is handled in `client/src/Main.tsx`. Furthermore the admin page has one route for each of the tabs which helps when reloading the site to select the previously selected tab. There is also a route for logging in with code which makes it possible to go to for example `localhost:3000/CODE123` to automatically join a competition with that code.

3.2.4 Authentication

Authentication is managed by using JWT from the API. The JWT for logging in is stored in local storage under `token`. The JWT for active presentations are stored in local storage `RoleToken` so for example the token for Operator is stored in local storage under `OperatorToken`.

3.2.5 Prettier and ESLint

`ESLint` is used to set rules for syntax, `prettier` is then used to enforce these rules when saving a file. ESLint is set to only warn about linting warnings. These libraries have their own config files which can be used to change their behavior: `client/.eslintrc` and `client/.prettierrc`

3.2.6 Redux

`Redux` is used for state management along with the `thunk` middleware which helps with asynchronous actions. Action creators are under `client/src/actions.ts`, these dispatch actions to the reducers under `client/src/reducers.ts` that update the state. The interfaces for the states is saved in each reducer along with their initial state. When updating the state in the reducers the action payload is casted to the correct type to make the store correctly typed.

3.2.7 Interfaces

In `client/src/interfaces` all interfaces that are shared in the client is located. `client/src/interfaces/ApiModels.ts` and `client/src/interfaces/ApiRichModels.ts` includes all models from the api and should always be updated when editing models on the back-end. This folder also includes some more specific interfaces that are re-used in the client.

3.3 Server overview

The server has two main responsibilities. The first is to handle API calls from the client to store, update and delete information, such as competitions or users. It also needs to make sure that only authorized people can access these. The other responsibility is to sync slides, timer and answers between clients in an active competition. Both of these will be described in more detail below.

3.3.1 Libraries

The server is built in [Flask](#). A few extensions to Flask are also used. [flask-smorest](#) is used to defined the API routes. It is this library that automatically documents the API on `localhost:5000` using [Swagger](#). [marshmallow](#) is used to convert database objects in JSON and to parse JSON back into Python objects. [SQLAlchemy](#) is used to interface with the SQL database that is used. More specifically [Flask-SQLAlchemy](#) is used to integrate it with Flask. [Flask-Bcrypt](#) is used to encrypt passwords.

3.3.2 Receiving API calls

An API call is a way for the client to communicate with the server. When a request is received the server begins by authorizing it (making sure the person sending the request is allowed to access the route). After that it confirms that it got all information in the request that it needed. The server will then process the client request. Finally it generates a response, usually in the form of an object from the database. All of these steps are described in more detail below.

Routes

Each existing route that can be called is specified in the files in the `app/apis/` folder. All available routes can also be seen by navigating to `localhost:5000` after starting the server.

Authorization

When the server receives an API call it will first check that the call is authorized. The authorization is done using JSON Web Tokens (JWT) by comparing the contents of them with what is expected. Whenever a client logs into an account or joins a competition, it is given a JWT generated by the server, and the client will need to use this token in every subsequent request sent to the server in order to authenticate itself.

The needed authorization is specified by the `ExtendedBlueprint.authorization()` decorator. This decorator specifies who is allowed to access this route, which can either be users with specific roles, or people that have joined competitions with specific views. If the route is not decorated everyone is allowed to access it, and the only routes currently like that is, by necessity, logging in as a user and joining a competition.

JSON Web Tokens (JWT)

JSON Web Tokens (JWT) are used for authentication, both for API and socket events. A JWT is created on the server when a user logs in or connects to a competition. Some information is stored in the JWT, which can be seen in the file `server/app/apis/auth.py`. The JWT is also encrypted using the secret key defined in `server/configmodule.py`. (NOTE: Change this key before running the server in production). The client can read the contents of the JWT but cannot modify them because it doesn't have access to the secret key. This is why the server can simply read the contents of the JWT to be sure that the client is who it says it is.

Parsing request

The server receives data in three ways: In the query string, body and header. The data in the body and header is sent in JSON format and needs to be converted into Python dictionaries. What data a route needs is specified by a marshmallow schema and blueprint from flask-smorest.

Handling request

After the request has been authorized and parsed the server will process the request. What it does depends on the route and the given arguments, but it usually gets, edits or deletes something from the database. The server uses an SQL database and interfaces to it via SQLAlchemy. Everything related to the database is located in the `app/database/` folder.

Responding

When the server has processed the request it usually responds with an item from the database. Converting a database object to json is done with marshmallow. This conversion is specified in two files in the folder `app/core/`. The file `schemas.py` converts a record in the database field by field. The file `rich_schemas.py` on the other hand converts an `id` in one table to an entire object in the another table, thus the name `rich`. In this way, for example, an entire competition with its teams, codes, slides and the slides' questions and components can be returned in a single API call.

3.3.3 Active competitions

Slides, timers, and answers needs to be synced during an active presentation. This is done using SocketIO together with flask-socketio. Sent events are also authorized via JWT, basically the same way as the for the API calls. But for socket events, the decorator that is used to authenticate them is `sockets.authorization()`. Whenever a client joins a competition they will connect via sockets. A single competition cannot be active more than once at the same time. This means that you will need to make a copy of a competition if you want to run the same competition at several locations at the same time. All of the functionality related to an active competition and sockets can be found in the file `app/core/sockets.py`. The terms *active competition* and *presentation* are equivalent.

Starting and joining presentations

Whenever a code is typed in to the client it will be checked via the `api/auth/code` API call. If there is such a code and it was an operator code, the client will receive the JWT it will need to use to authenticate itself. If there is such a code and the associated competition is active, the client will also receive a JWT for its corresponding role. Both of these cases will be handled by the default `connect` event, using the JWT received from the API call. The server can see what is stored in the JWT and do different things depending on its contents.

Syncing between clients

There are two other events that is used. The operator will emit the `sync` event to synchronise some values to all other clients connected to the same competition. The server will then send `sync` to all connected clients with the values that was updated. The server will also store these values and will synchronise these when a client joins a presentation. The operator can also emit `end_presentation` to disconnect all clients from its presentation. This will also end the presentation.

3.4 Database overview

INSTALLATION

This section will describe how to install the application. You will need to install both the client and the server.

4.1 Installing the client

It is recommended to use [Visual Studio Code](#) to install and use the client, but it is not necessary. In order to install the client, you will need to do the following:

Install [Node \(LTS\)](#).

Clone the git repository [teknikattan-scoring-system](#).

Open a terminal and navigate to the root of the cloned project.

Install all client dependencies:

```
cd client
npm install
```

You should now be ready to start the client. Try it by running `npm run start`. A web page should open where you can see the [login page](#). If you are using VS Code you can also start the client with the `task start client`.

4.1.1 If there are any errors, please try this.

```
npm rm react react-dom
npm i -s react react-dom
```

4.2 Installing the server

The steps to install the server depend on if you are on Windows or Linux and if you are a developer or are running it in production.

4.2.1 Windows

Clone the Git repository:

```
git clone https://gitlab.liu.se/tddd96-grupp11/teknikattan-scoring-system
cd ./teknikattan-scoring-system/
```

Install Python.

Make sure Python is *installed properly*.

Make sure pip is *installed properly*.

Install virtualenv and create a virtual environment:

```
pip install virtualenv
cd server
py -m venv env
```

Activate the virtual environment:

```
Set-ExecutionPolicy Unrestricted -Scope Process
./env/Scripts/activate
```

Continue to *development and production*.

4.2.2 Linux (Ubuntu)

Clone the Git repository:

```
git clone https://gitlab.liu.se/tddd96-grupp11/teknikattan-scoring-system
cd ./teknikattan-scoring-system/
```

Install Python.

Make sure Python is *installed properly*.

Install pip:

```
sudo apt install python3-pip
```

Make sure pip is *installed properly*.

Install and create a Python virtual environment and activate it:

```
sudo apt install python3-venv
cd server
py -m venv env
source env/bin/activate
```

Continue to *development and production*.

4.2.3 Development and production

Which dependencies you install will depend on if you are a developer or running the server in production.

If running in production:

```
pip install -r requirements.txt
```

If you are a developer:

```
pip install -r requirements-dev.txt
```

You should now be ready to start the server. Try it by running `py main.py` and navigate to `localhost:5000`. If everything worked as it should you should see a list of all available API calls. If you are using VS Code you can also start the server with the `task start server`.

4.2.4 Common issues

If you have any issues while installing, some of the things below might help.

Running Python

Test that Python is installed properly:

```
py --version
```

Make sure Python version > 3. If it works, you should see something along the lines of:

```
Python 3.9.4
```

If `py` is not working, try one of the following instead:

```
py -3
py3
python
python3
```

Running pip

Test that `pip` is installed properly:

```
pip --version
```

Make sure `pip` is running with Python 3.x (not Python 2.x). If everything works, it should look something along the lines of:

```
pip 20.2.3 from d:\home\workspace\technikattan-scoring-system\server\env\lib\site-
-packages\pip (python 3.9)
```

If `pip` is not running with Python 3.x, try one of the following instead:

```
pip3
py -m pip
```

If you still have trouble, try this [guide](#).

Problem: Failed building wheel for <package> when calling pip

Run the following command before installing the requirements:

```
pip install wheel
```

This [guide](#) can help you troubleshoot this problem further.

Problem: psycopg

```
pip install psycopg2
```

DEVELOPMENT

This section will give all the instructions necessary to continue the development of this project. Some recommendations for how to go about it will also be given.

5.1 Frontend

Here it is described how to work with the frontend in the system.

5.1.1 Working with TypeScript

The main programming language used for the front end is TypeScript.

npm

npm is the node package manager. Below we briefly describe how to use it. All of the following snippets assume you are in the `client` folder.

To install a module, run `npm install <module>`.

To uninstall a module, run `npm uninstall <module>`.

To install all project dependencies, run `npm install`.

It is important to remember to install the project dependencies whenever someone else has added new ones to the project.

5.2 Backend

5.2.1 Arguments when running backend

When running `main.py` several arguments can be used

```
arg1(action):    server(default), populate
arg2(mode):      dev(default), prod
arg3(database):  lite(default), postgre
```

Running server

```
main.py -> same as below  
main.py server dev lite -> Run server in dev-mode with sql-lite  
main.py server prod postgre -> Run server in production-mode with postgresql
```

Populating backend

```
main.py populate dev lite -> Populate database in dev-mode with sql-lite  
main.py populate prod postgre -> Populate database in production-mode with postgresql
```

5.2.2 Working with Python

In this section we briefly describe how to work with Python.

Virtual environments

Python virtual environments are used to isolate packages for each project from each other. When *installing the server* you installed `virtualenv` and created and activated a virtual environment.

Pip

Python uses `pip` to manage its packages. Here we briefly describe to use it. All of the following instructions assume you have created and activated a virtual environment and are located in the server folder.

To install a package, run `pip install <package>`.

To uninstall a package, run `pip uninstall <package>`.

To save a package as a dependency to the project, run `pip freeze > requirements.txt`.

To install all project dependencies, run `pip install -r requirements.txt`.

Remember to install the project dependencies whenever you or someone else has added new ones to the project.

5.3 Visual Studio Code

The development of this project was mainly done using Visual Studio Code (VS Code). It is not that surprising, then, that we recommend you use it.

5.3.1 Extensions

When you first open the repository in Visual Studio Code it will ask you to install all recommended extensions, which you should do. We used a few extensions to help with the development of this project.

The Python and Pylance extensions help with linting Python code, auto imports, syntax highlighting and much more.

Prettier is an extension used to format JavaScript and TypeScript. ESLint is used to lint JavaScript and TypeScript code.

Live Share is an extension that is used to write code together at the same time, much like a Google Docs document. There were however a few issues with the Python extension that made Live Share hard to work with.

5.3.2 Tasks

A task in VS Code is a simple action that can be run by pressing `ctrl+shift+p`, searching for and selecting `Tasks: Run Task`. These tasks are configured in the `.vscode/tasks.json` file. Tasks that are marked as build tasks (starting and testing tasks as well as populate) can also be run with `ctrl+shift+b`. A few such tasks has been setup in this project and will be described below.

The `Start server` task will start the server.

The `Start client` task will start the client.

The `Start client and server` task will start both the client and the server.

The `Populate database` task will populate the database with a few competitions, teams, users and such. Look in the `populate.py` to see exactly what it adds. Remember to always run this after changing the structure of the database.

The `Test server` task will run the server tests located in the `server/tests/` folder.

The `Open server coverage` task can only be run after running the server tests (`Test server` task) and will open the coverage report generated by those tests in a web browser.

The `Unit tests` task will run the unit tests for the client.

The `Run e2e tests` task will run the end-to-end tests.

The `Open client coverage` task can only be run after running the client tests (`Unit tests` task) and will open the coverage report generated by those tests in a web browser.

The `Generate documentation` task will generate the project documentation, i.e. this document, in the `docs/build/html/` folder.

The `Open documentation` task can only be run after generating the documentation and will open it in a web browser.

5.4 External programs

These are some useful programs that can help with the development.

5.4.1 Postman

[Postman](#) is a program used to test API calls. You can create and edit API calls, change the body, headers and even share what you have saved. It's very helpful when developing APIs.

5.4.2 DB Browser for SQLite

[DB Browser for SQLite](#) is used to see what is currently stored in the database. You can even edit values.

5.5 Further development

Because the project was time limited a lot is left to be done. Below we will give two different types of things to improve. The first type is functionality, bugs and aesthetics which improves the usability of the system. The second type is refactoring which is basically just things related to the source code. This won't effect the end user but will certainly improve the system as a whole.

5.5.1 Functionality, bugs and aesthetics

Most of the basic functionality of the system is already completed. There are however a few major things left to be done.

Different question types

The system needs to support a lot of different types of questions. A list of all the questions that needs to be supported (and more) can be found on [Teknikattan scoring system](#).

Scaling of components

Components rendered in `SlideDisplay.tsx` in client are scaled inconsistently and should use [scale transform](#) from CSS.

5.5.2 Refactoring

Here we will give a list of things we think will improve the system. It is not certain that they are a better solutions but definitely something to look into.

Server configuration

The server can be configured to run in development or production mode and can use either `sllite` or `postgresql`. The code to handle these configuration options were written very late in the project and should be refactored, maybe using `argparse`.

TESTING

Here we briefly describe how we have tested the system. Both unit tests for the client and server has been made. Some end-to-end tests have also been made that tests both the server and client at the same time.

6.1 Testing the client

The clients tests are the files named `<name>.test.ts`. They test the file called `<name>.ts`. They are run using the *VS Code task* `Unit tests`.

6.2 Testing the server

The Python testing framework used to test the server is `pytest`.

The server tests are located in the folder `./server/tests`. The tests are further divided into files that test the database (`test_db.py`) and test the api (`test_api.py`).

The file `test_helpers.py` is used to store some common functionality between the tests, such as adding default values to the database. There are also some functions that makes using the api easier, such as the `get`, `post` and `delete` functions.

Run the tests by running the *VS Code task* `Test server`. After that you can see what has been tested by opening the server coverage using the task `Open server coverage`.

6.3 End to end tests

The end to end tests are tests that test the entire system, both the server and the client. They are stored in the folder `/client/src/e2e/`. Both the client and the server need to be running for the end to end tests to work. The tests are run using the *VS Code task* `Run e2e tests`.

DOCUMENTATION

Here we describe how to generate this entire web page. We also describe how to generate documentation for the client and server modules.

7.1 Generating this document

To generate this document you need to do a few things.

You will need to install `make`. If you are on Linux you probably already have it installed and can skip the two following steps. If you are on Windows you need to do the following:

Download and install [Chocolatey](#).

Install `make` using Chocolatey (open PowerShell as administrator):

```
choco install make
```

You also need to *install the server*.

You should now be able to generate the documentation by activating the Python virtual environment, navigating to `docs/` and running `make html`. Alternatively you can also run the [VS Code task](#) `Generate documentation`, which will do the same thing. If everything went well you should be able to open it by running (from the `docs/` folder) `start ./build/html/index.html` or running the task `Open documentation`, which does the same thing.

7.2 Generating documentation for the client

To generate documentation for the client you first need to *install the client*.

After that you will be able to generate the documentation by running:

```
cd client/  
typedoc
```

You will then be able to open it by running:

```
start ./docs/index.html
```

If you want to include the documentation from the tests, go to the file `client/tsconfig.json` and comment out the line `"exclude": "**/*.test.*"`.

7.3 Generating documentation for the server

Generating the server documentation involves a few more steps.

You need to follow the same preparatory steps as you did to *generate this document*. That is installing make and installing the server.

Run the following:

```
cd server/  
mkdir docs  
cd docs/  
sphinx-quickstart
```

You will be asked a few questions about how to configure Sphinx. Just press enter on all, which will use the default. You can enter the correct project name and/or author if you want, but it's not necessary, no one but you will see it anyway.

Then will need to modify a few files. First add the following code snippet after the first block of comments, above the “project information” comment, in the file `./server/docs/conf.py`:

```
import os  
import sys  
  
basepath = os.path.dirname(__file__)  
filepath = os.path.abspath(os.path.join(basepath, "../"))  
sys.path.insert(0, filepath)
```

Then in the same file add an extension to the list of extensions, like so:

```
extensions = ["sphinx.ext.autodoc"]
```

Then just write the word app on line 13 in the file `server/docs/index.rst`. The file will then look something like:

```
.. toctree::  
    :maxdepth: 2  
    :caption: Contents:  
  
    app
```

Then the documentation can be generated by running (still in the docs/ folder):

```
sphinx-apidoc -o ./ ../app --no-toc -f --separate --module-first  
make html
```

You can then open it by typing:

```
start ../_build/html/index.html
```

You could add all of the files we just added, except `_build/`, to Git if you want.

CONTACT

The people involved in the project, their email, their role in the project, what they worked on generally and, if anything, they worked on most will be described below. Please feel free to contact us if you have any questions.

Name	Email	Role	Generally	Especially
Albin Henriks-son	albhe428@student.liu.se	Test Leader	Front end	Presentations, Editor, Test- ing
Sebastian Karls-son	se- bka991@student.liu.se	Architect	Front end	Editor, Uploading pictures
Victor Löfgren	vi- clo211@student.liu.se	Configuration Manage- ment	Back end	Documentation, Sockets, API
Björn Modée	bjomo323@student.liu.se	Quality Assurance	Front end	Redux
Josef Olsson	josol381@student.liu.se	Team Leader	Back end	Database, Testing
Max Rüdiger	maxru105@student.liu.se	Document Management	Front end	
Carl Schönfelder	carsc272@student.liu.se	Development Leader	Back end	Database, Uploading pic- tures
Emil Wahlqvist	emiwa210@student.liu.se	Analyst	Front end	Editor

LICENSE**MIT License**

Copyright (c) 2021 Linköping University, TDDD96 Grupp 1

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.