

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

# TeamCalendar

*Team ProgExTRAORDINAIRE:  
Klejdi Galushi, Marc Roemer, Felix Schneider*

Supervisor:  
Salvatore Sabba

June 12, 2024

# Contents

<b>1</b>	<b>Target determination</b>	<b>2</b>
1.1	Mandatory criteria . . . . .	2
1.2	Target criteria . . . . .	2
1.3	Optional criteria . . . . .	2
1.4	Exclusion criteria . . . . .	2
<b>2</b>	<b>Product deployment</b>	<b>3</b>
2.1	Application area . . . . .	3
2.2	Target groups . . . . .	3
2.3	Operating conditions . . . . .	3
2.4	Development setup . . . . .	3
<b>3</b>	<b>Product overview</b>	<b>3</b>
<b>4</b>	<b>Product functions</b>	<b>4</b>
<b>5</b>	<b>Product data</b>	<b>5</b>
<b>6</b>	<b>Product services</b>	<b>5</b>
<b>7</b>	<b>Quality requirements</b>	<b>5</b>
<b>8</b>	<b>User interface</b>	<b>5</b>
8.1	Access rights . . . . .	7
<b>9</b>	<b>Non-functional requirements</b>	<b>7</b>
<b>10</b>	<b>Technical product environment</b>	<b>7</b>
10.1	Software . . . . .	7
10.1.1	Client . . . . .	7
10.1.2	Server . . . . .	7
10.2	Hardware . . . . .	7
10.2.1	Server . . . . .	7
10.2.2	Client . . . . .	8
10.3	Orgware . . . . .	8
10.4	Product interfaces . . . . .	8
<b>11</b>	<b>Special requirements for the development environment</b>	<b>9</b>
11.1	Software . . . . .	9
11.2	Hardware . . . . .	10
11.3	Orgware . . . . .	10
11.4	Development interfaces . . . . .	10
<b>12</b>	<b>Division into sub-products</b>	<b>10</b>
<b>13</b>	<b>Additions</b>	<b>11</b>
<b>14</b>	<b>Glossary</b>	<b>11</b>

# **1 Target determination**

## **1.1 Mandatory criteria**

- Program consists of front end, back end and database
- Users can be changed once chosen
- Calendars can be shared
- Calendars have an owner
- Calendar ownership can be transferred
- Calendar events can be added, edited and removed
- Calendar events have a start and end time
- User will get notified at a custom time and at the start of the event

## **1.2 Target criteria**

- Readable text
- Multiple custom notifications

## **1.3 Optional criteria**

- Exclusive calendar events, that reserve a certain time slot so that calendar events can not overlap
- Warn user if creating a calendar event will result in overlapping calendar events
- Show next available time slot when creating a new calendar event
- Add users in the front end
- Delete users in the front end
- Email notifications
- Change calendar event color

## **1.4 Exclusion criteria**

Viewing the calendar of other users will not be implemented as changing a user is possible.

## 2 Product deployment

### 2.1 Application area

This tool should be used by teams or individuals to organize themselves by providing a timetable. For this purpose team calendars can be shared and each user will get shown an overview of all calendars they and their team is responsible for.

### 2.2 Target groups

The target groups for this tool are:

- Students
- Companies
- Sport clubs
- Institutions

Based on this it is assumed that the frontend needs to be readable by people of all age groups.

### 2.3 Operating conditions

The Program should be deployed on a single server that is running 24/7 in the cloud or on premises for each company. The program should not need be to administrated once deployed.

### 2.4 Development setup

1. Create a venv in `./Backend` using `requirements.txt` with `pip install --upgrade -r requirements.txt`
2. Activate venv
3. Navigate to `mysite` and run development server with `python manage.py runserver`
4. Execute `npm install` in `./Frontend`
5. Navigate to `my-app` and run development server with `npm start`

## 3 Product overview

Upon first visiting the website, a user selection interface will be shown where a user can be selected. After selecting a user, a calendar will be dynamically generated, displaying all of the calendar events that the user has access to. User can be changed through the Sidebar. Additionally, the site offers a dedicated group management window for managing groups.

## 4 Product functions

- User Selection

The website starts with a user selection page, where the user can choose an user out of a list of users.

After the user selection, the user will gain access to their own instance of the calendar.

- Calendar Interface

The user can change the displayed calendar month by clicking the "<" or ">" button.

Alternatively, the user can go back to the current month by clicking on the "today" button.

- Event Handling

Calendar events can be created by clicking on an empty part of a day grid.

When clicking on an event, a popup window will show, where the user can modify the title name, start time, end time, choose a group and add/remove reminders. The changes will be applied after pressing OK.

Calendar events can be moved by clicking and dragging them with the mouse.

Calendar events can be deleted by clicking on the "DELETE" button in the popup window.

- Menu Sidebar

The user can open a menu sidebar by clicking on a button on the top left of the page.

In the menu sidebar, the user will be able to see their username, the groups they are in and an option to log out.

- Group Handling

The user can create a group after clicking on "Create Group" below the group names on the menu sidebar.

After clicking on a group name in the menu sidebar, the admin can modify the group name, add/delete group members or leave the group, whereas other group members can only view the group or have the option to leave.

- Reminders

When an event's reminder time is reached, the user will get a notification from the website.

## 5 Product data

To be saved Data will include:

- Users
  - Name
  - Email
- Group name
- Calendar events consisting of:
  - Title
  - Description
  - Start Time
  - End Time
  - Reminder Times
- Users with access to calendar

## 6 Product services

The program should generate the calendar view in a reasonable time and all calendar events have to be correct.

## 7 Quality requirements

It is required that the software implements all Mandatory criteria, and responds to the user in a reasonable time. The software should be easy to understand, maneuverable and text should be big enough to be read. Once deployed it should work without needed intervention from a software developer. If needed the server should be able to be scaled up vertically to accommodate more traffic and data. The client side should be accessible for all users with an up to date web browser that is able to run react 18.3.0 web apps or newer.

## 8 User interface

The user interface contains three main views:

1. The user selection page

This is the page the user first sees when opening the website.

In the middle, they can see the website's logo, a drop-down list and a "select" button which allows them access to the website.

If the user clicks on the drop-down list, all available usernames will be shown.

By default, the "select" button will be grayed out, but after choosing an username, it will switch colors, signaling the user that they are ready to access the calendar page.

2. The calendar page

This is the main page of the website.

On the top, from left to right, the user can find:

A button to activate the menu sidebar

The website's logo

A text welcoming the user (e.g. "Welcome, user!")

The calendar is located in the middle of the website, where the user can interact with it to create/modify/delete events and change which calendar month they would like to display.

By default, the calendar displays the current month and the current day will have a yellow background.

If the user decides to create an event, a pop-up window will show up, containing:

Text boxes for the event name and start and end time

A checkbox list for choosing groups

A dynamic list for adding/removing reminders

The bottom side of the website only contains the year of creation and our team's name.

3. The menu sidebar

When clicking on the top-left button in the calendar page, the calendar page will fade out and a column will appear on the left hand side of the website.

From top to bottom, this sidebar contains:

The username of the current user

The list of groups the user has joined/created. By clicking any group name, the website will show a pop-up window with the group name, members and options depending on whether the user is an admin or not.

A log-out option, where the user can log out and leave the calendar page, going back to the user selection page.

## 8.1 Access rights

After selecting a user on the user list, the user has access on their own instance of the calendar and no other instances. However, the user's events can be shared among the groups they have joined.

Inside groups, users with the admin role have exclusive rights to modify the group name and add/remove members.

## 9 Non-functional requirements

The saved data shall not be accessible for 3rd parties. The data shall only be used for the intended purposes of providing a timetable and reminders. It shall not be used by anyone that has access to them for any purpose other than what the user consented to. The data shall not be sold or be used for personalized advertisements. The data shall be persistent and encrypted.

## 10 Technical product environment

### 10.1 Software

#### 10.1.1 Client

- Any browser that supports react 18.3.0 web apps or newer

#### 10.1.2 Server

- python (3.12.2)
- Django (5.0.4)
- django-filter (24.2)
- djangorestframework (3.15.1)
- SQLAlchemy (2.0.30)
- MySQL



## **10.2 Hardware**

### **10.2.1 Server**

Linux or Windows server with internet access.  
Mac not tested.

### **10.2.2 Client**

Any device capable of running a browser that supports react 18.3.0 web apps or newer with internet access.

## **10.3 Orgware**

The deployed code needs to be documented to make servicing it possible. Intended documentation is the GitHub project where developers made comments for their decisions, Mask Draft and an Entity-relationship diagram for the database. Once deployed the tool should not need to be administrated.

## **10.4 Product interfaces**

The frontend and backend need to be connected via their ip addresses in a network.

The backend and database need to be connected via their ip addresses in a network.

External interfaces are not needed.

## 11 Special requirements for the development environment

### 11.1 Software

- Git for downloading the project
- GitHub for accessing the project
- PyCharm for editing the backend and database
- Visual Studio Code for editing the frontend and backend
- python (3.12.2)
- nodejs (20.12.2)
- @fullcalendar/core (6.1.11 or newer)
- @fullcalendar/daygrid (6.1.11 or newer)
- @fullcalendar/interaction (6.1.11 or newer)
- @fullcalendar/react (6.1.11 or newer)
- @testing-library/jest-dom (5.17.0 or newer)
- @testing-library/react (13.4.0 or newer)
- @testing-library/user-event (13.5.0 or newer)
- axios (1.6.8 or newer)
- react (18.3.0 or newer)
- react-dom (18.3.0 or newer)
- react-scripts (5.0.1 or newer)
- web-vitals (2.1.4 or newer)
- asgiref (3.8.1)
- Django (5.0.4)
- django-filter (24.2)
- django-rest-framework (3.15.1)
- Markdown (3.6)
- sqlparse (0.5.0)
- SQLAlchemy (2.0.30)
- MySQL

## 11.2 Hardware

Windows or Linux computer capable of running all software outlined in 11.1  
Software with internet access.  
Mac not tested.

## 11.3 Orgware

The agile software development method will be used. GitHub will provide all tools necessary for agile development. Sprints last a week and end on Thursdays in the lecture.

GitHub is used for:

- Storing code
- Managing code
- Sharing code
- Version control
- Continuous Integration/Testing (GitHub actions)
- Access control
- Bug tracking (Issues)
- Task management (GitHub projects)

## 11.4 Development interfaces

The front end and back end need to be connected via their ip addresses in a development environment.

The back end and database need to be connected via their ip addresses in a development environment.

The database is hosted externally at <http://descus.de>.

## 12 Division into sub-products

The product is layered and divided into 3 parts according to a layered architecture. The top layer is the front end providing an interface to the end user. The second layer is a back end that provides an endpoint for the front end to access data. The third and last layer is the database where the data is persistently saved. It is connected to the back end via ip address and password.

The front end is written in react with nodejs and javascript.

The back end is written in django using python3. The database is written using SQLAlchemy and MySQL.

## 13 Additions

The Functional Specification Document shall be delivered by 12.06.2024

The Mask draft shall be delivered by 19.06.2024 at the latest

The database concept diagram shall be delivered by 26.06.2024 at the latest.

The final product shall be delivered by 10.07.2024 at the latest.

## 14 Glossary

**Front end:** A front end is the part of the software that the user interacts with, providing all of the functionality of the program.

**Back end:** A back end is the server side component of the program, it is enabling all of the functionality. In this case by connecting to the database and providing the data to the front end to be displayed in the right format.

**venv:** A venv is a virtual environment provided by python. It is needed so that all required packages and dependencies are installed for the project after setting it up, rather than installing all packages system wide by hand.