REPORT OF FUTURE.IO

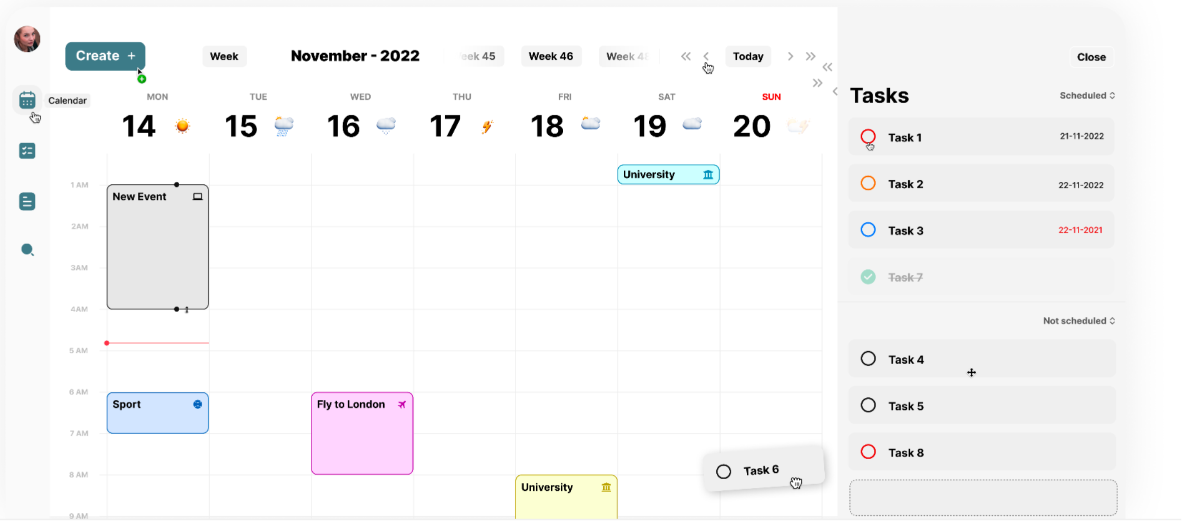
Elia Jabour, Hugo Krul, Pepijn Uuldriks

Informatica introduction project

Coach: Mahsa Ghanavatinasab

[16-12-2022]

Future.io



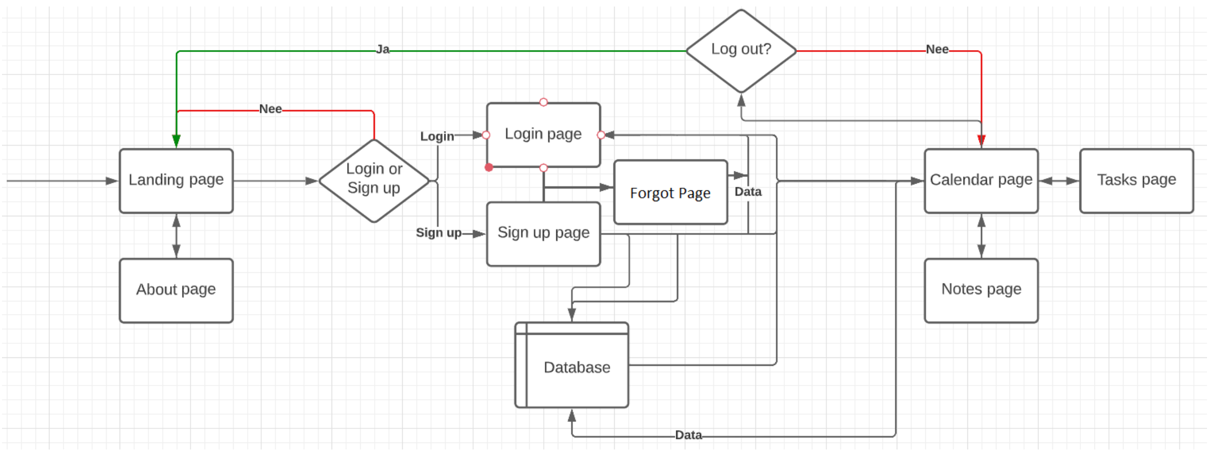
Utrecht University

Table of contents

You can view our code here: https://github.com/Elia-J/FutureWebsite
The readme inside provides instructions on how to run and view our code.

Updated Vision (the relevant parts)

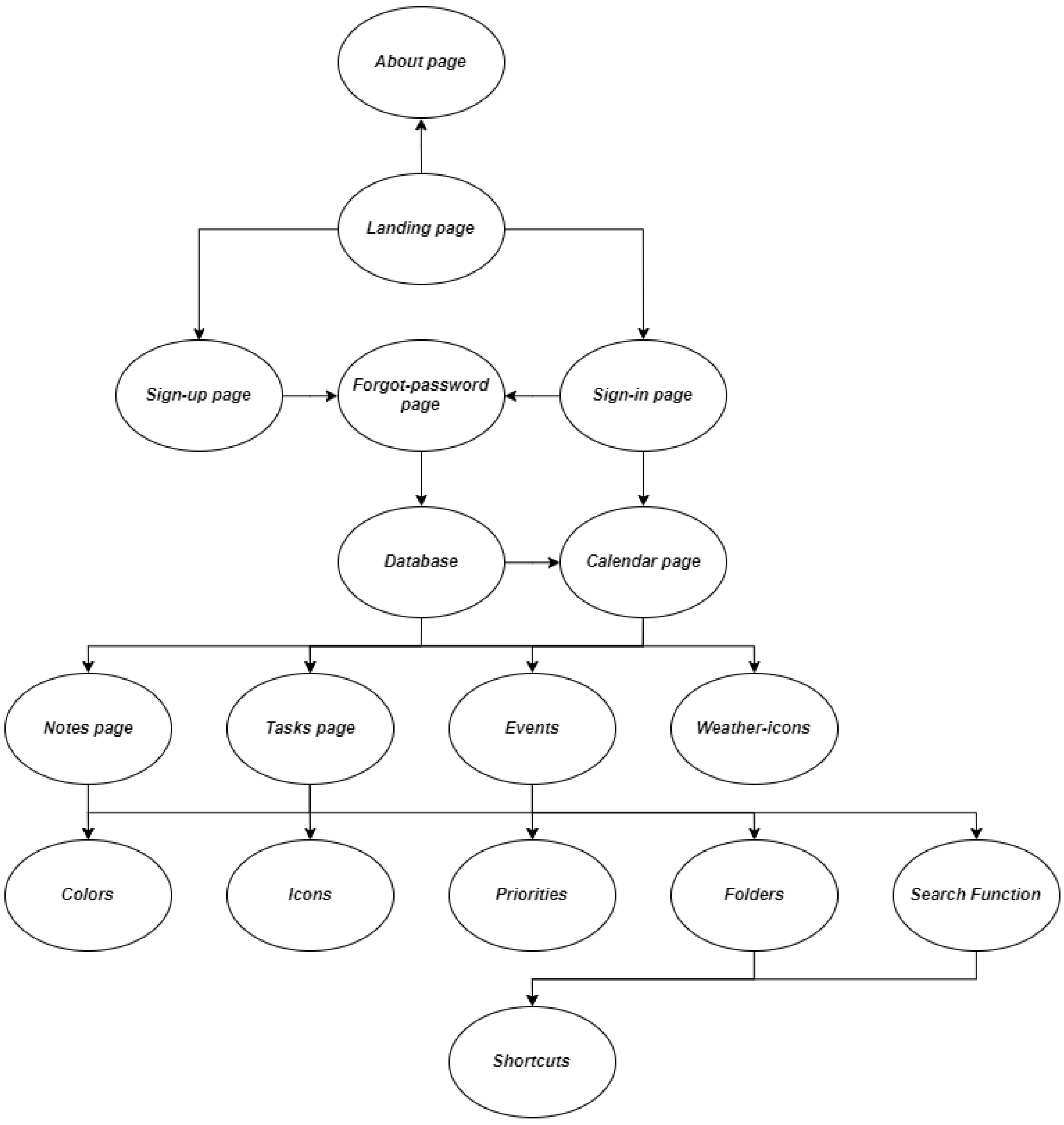
We have added a forgot-password page to our project and to reflect that we have updated our general structure image:



We have slightly changed the order in which we will implement things, the new order is as follows:
landing page → about page → login page → sign up page → sign-in page → forgot-password page → database → a calendar → tasks page → notes page→ events → weather-icons → search → priorities → folders → colours (for events, tasks, notes and folders) → icons (for events, tasks, notes and folders) → shortcuts → autosave *implementation → optimisation → ai*

We have added a forgot-password page to this order and we previously had a dark mode/light mode at the end of our implementation but we have instead decided to make every page compatible with such modes while making them.
We have added the feature autosave because it is tedious to constantly hit the save button if you want to save your notes. There are also a few refinements that need to be done on the different pages we have already made. If we have time left, we will try to optimize our code and make the website faster and easier to use. At the end of our run, if we still have time left, we would like to implement an AI which would help users with notes and tasks.

Our new dependency graph, with the added forgot-password page and with the dark/light mode simply becoming part of every page, looks like this:



We have also switched the minimum skeleton and minimum prototype, since it clearly doesn't make much sense to have a stricter minimum skeleton than a minimum prototype.

Minimum skeleton:
Our minimum skeleton will contain the front end of the website for all the pages, this includes dark mode/light mode. It should also be dynamic and be compatible with both mobile and desktop. It should give a position to all of its components, without giving functionality. One exception to this is the login functionality. This functionality will set up the database in which we will save the data of the calendar, notes and tasks and if that is set up, that means the implementation of the other data will be easier. That is why we choose to implement this feature in the minimum skeleton. We expect and hope that the above will be finished on the 2th of December.

Minimum Prototype:
The bare minimum prototype must contain the following: It must contain a landing page, so a user will have a page to go to when entering our website. It must contain a login/signup page, so a user can create an account and can subsequently login to this account. It must contain an about page, on which a user can find out who made the website and on which certain questions a potential user may have will be answered. The about page must also have a way for a user to contact us. It must have a calendar page which should have the following functionalities: a calendar on which a user can move between the weeks of the year and there must be a button which would allow a user to make events. Events must contain the following elements: a name, an optional description and two time-slots, including the date of these time-slots, between which the event takes place. Created events should be saved individually for each user in a database and viewable on a users personal calendar. It must contain an event page which would have the following functionalities: a user must be able to create and discard tasks on said page. Created tasks should contain the following elements: a name, an optional description and an optional deadline for the task. Created tasks should be saved individually for each user in the database and should be viewable from the task page. It should contain a note-page which should have the following functionalities: A way for a user to create and discard notes. Created notes should contain the following elements: a name, an optional description in which a user can type text and make said text bold, italic, etc. Created notes should be saved individually for each user in the database and should be viewable from the notes page.

The minimum prototype will be finished by the 9th of December. On the 14th of December, we will have a presentation about our project, so it will be good to have our minimum prototype ready before then.

Current status

We have made some good progress in the first half of this project and we have more than delivered on our minimum prototype. Currently our project contains the following pages: The landing page, the about page, the sign-in page, the sign-up page, the forgot-password page, the calendar page, the notes page and the tasks page. We have also made a database in supabase which contains the following tables: **profiles**, **events**, **tasks** and **notesv2**.

In **profiles** the following rows are present: *id* which is a unique id for every user and automatically gets generated (the primary key for this table), *updated_at* in which the timestamp at which a user updates their profile automatically gets recorded, *username* which is a the name a user can give themselves*, full_name* which records the full name of a user, *avatar_url* which contains the url to an image a user will eventually be able to upload to our database, *website* in which a users inputted website gets saved, *mode* in which gets recorded which mode (either light/dark/system) a user prefers and *timezone* in which a users timezone can be saved if a users inputs said timezone. In **events** the following rows are present: a unique *id* for every event (the primary key for this table), a *title*, *created_at* in which it gets recorded when an event is created, a *beginDate* in which it gets recorded when an event is supposed to start (date and time), a *endDate* in which it gets recorded when a event is supposed to end (date and time) and a linked *user_id* in which it gets recorded which user made an event (foreign key). In **notesv2** the following rows are present: *id* which is a unique id for every note (the primary key for this table), *created_at* which records when a note is created, *title* which records the title of a note, *description* which contains the textbody of a note and a linked *user_id* which records which user created a note (foreign key). In **tasks** the following rows are present: a unique *id* for every task, a linked *user_id* which records which user created a particular task (foreign key), a *inserted_at* which records the time and date in which a task is inserted in the table, a *title* which records the title of a task, a boolean *checked* which records if a task is completed or not, a *date* which records a deadline-date for a task, a *time* which records a deadline-time for a task and a *description* which contains a description for a task.

A couple of things are present on every page: all pages are compatible with dark/light/system mode and all pages work on both desktop and mobile.

The landing page is the first page a user sees when they visit our website. On this page there is a pleasant looking layout to welcome any potential user alongside our slogan: "The most powerful web app for *planning, organizing and scheduling* your life". A user can do the following things on this page: click on the themes button to change the theme of our website to either: light mode, dark mode or system mode. System mode in this instance simply sets the mode of our website to whatever mode a user's system is on (so light or dark). There is also a navigation bar on which there is a logo button, which would take a user to the landing page, an about button, which would take a user to the about page, a sign-up button, which would take a user to the sign-up page and a sign-in page which would take a user the the sign-in page.

The navigation bar described above is also used on the about page. On the about page there is a picture on which the three of us stand. Directly below this picture there is a caption indicating who is who in the picture. Below this there is a frequently asked question section.

When a user clicks on a question, an answer to the question will appear beneath it, the animation underlying this acts as an accordion. On the bottom of the page there is a way for a user to send us a message.

On the sign-up page a user can create an account and on the sign-in page a user can login to an already created account. A user can login by inputting their email and password, but they can also sign in with Magic Link or Google. On the forgot-password page a user can reset their password using their email.

When a user signs in, they will be sent to the calendar page. On the calendar page there is, as the name suggests, a calendar. This calendar can be viewed in day, week or month mode. The calendar shows via visual reference which date it currently is and it also shows via visual reference what time it is. A user can move between days, weeks or months by clicking on the arrow buttons and return to the present day by clicking on the ‘Today’ button. On top of the calendar it shows what date(s) and week is currently being shown. In the top left corner of the page there is a create button which will eventually allow a user to create an event. An created event will be saved in our database as described above. Currently an event can be created by dragging down the mouse on the calendar itself. The calendar page, the notes page and the tasks page make use of the same navigation bar, which would allow a user to go to said pages and also open up the user settings. There is also a magnifying glass icon on the navigation bar, which currently has no functionality, but which will eventually serve as a search button.

On the notes page a user can create, view, edit and delete notes. One can create a note simply by inputting a title and then clicking on the create note button. Then you can view your newly created notes but also your older notes by clicking on the note title. This click would load in your note, in which you can edit the text body. Text can be manipulated by making it **bold**, *italics,* <u>underline</u>, </Code>, Header1, Header2, “Quote”. You can also create bullet points and align your text left, center or right. When a user has edited their note to their liking they can save it by clicking on the save button. A note can be deleted by clicking on the trashcan-icon accompanying each note.

On the tasks page there are two main sections: the scheduled tasks section and the non-scheduled tasks section. In the scheduled tasks section all tasks which have a deadline get displayed. Non-scheduled tasks are simply those tasks which have no deadline. A user can create a task by clicking on the *‘Create +’* button and edit an already existing task by clicking on the title of one of the tasks. When a user clicks on one of these a form will appear in which one can input a required task title, an optional date, an optional time or an optional description. In this form one can also click on the ‘x’ button to click the form away, or one can save the task by clicking on the save button or one can discard a task by clicking on the discard button. Scheduled tasks can be sorted in alphabetical order (a-z), alphabetical order (z-a), date ascending or date descending. Non-scheduled tasks can be sorted in alphabetical order (a-z) or alphabetical order (z-a). One can change the manner in which the tasks are sorted by way of a combobox. A user can ‘check’ or ‘uncheck’ a task by clicking on the check button accompanying each task. Checked tasks immediately get dropped down their respective list.

When a user clicks on their profile picture a dropdown appears in which a user can access the user-settings, change the mode in which the website is viewed (light mode/dark mode/system mode) or sign out. In the user-settings there are two sections: the account section and the general section. In the account section a user can change or delete their profile picture, change their full name, change their username, change their website, change their password or delete their account. In the general section a user can change the timezone (which is used to display the time in the calendar), change the first day of the week (becomes the first day of the week in the users calendar), change the theme between light, dark and system and synchronize the theme with their account, so that it is applied to all devices.

Planning

Status:
📌Planned
💪in Progress
✅Completed

Week	Tasks
Week 1	✅Vision - General Description - Use Cases - Features - Project components - Rough planning - Minimum skeleton - Minimum prototype - Challenges - User Interface sketch - Tools - Weekly Workflow - Code Review - Risk Mitigation ✅Created a WhatsApp group ✅Created a Notion workspace ✅Created a supabase account & new project ✅Created a prototype in Figma - Designed the logo - Designed the sign in, sign up, password reset and forgot you password pages - Designed the calendar and the side panel - Designed the settings
Week 2	✅Development environment - Installed node-js. - Connected to gitlab. ✅Created Hello world Next-js project ✅Connected Next-js with our database. ✅Built sign in, sign out, sign up, password reset and forgot you password pages and/or functions. ✅Added the ability to sign in with Magic link and google account. ✅Shared layout is added - Same layout across different pages. ✅Dark & light mode is added ✅Created the Landing page - Toggle for the theme - Navbar component ✅Created the About page - Big image - Description - Contact form.
Week 3	✅Calendar 📅 - Installed the packages needed to create a base calendar.

	- Built the week view calendar. ✅Built the Settings Popup - Where the user can change his profile picture, full name, username, password, theme and website URL. ✅Built a dropdown menu - the option to go to settings. - the option to change the theme. - the option to sign out. ✅Built the left navbar - where the user can navigate between calendar, tasks and notes. ✅Notes 📗 - Built notes as dynamic pages - Build the notes layout - Get notes from the database. ✅Tasks 📝 - simple load task form the database
Week 4	✅Added day view and month view to the calendar. ✅Added calendar buttons such as: next week, next two weeks, today, previous week , previous two weeks. ✅Added red line to indicate the current time. ✅Loading events from the database to the Calendar. ✅Save events from the calendar to the database. ✅Added the right tasks panel. ✅Added editor to edit and save notes. ✅Added css color palette ✅Added checkbox to check tasks ✅the user can see the scheduled and the non scheduled tasks separated from each other.
Week 5	✅Improved the calendar. - Added week number. - Calendar height is improved ✅Improved the Notes. - ✅Improved the Tasks. - Tasks are now as notes dynamic ✅Midterm Presentation - Created a PowerPoint - Live demo ✅Code - Merge the code from different branches to the main branch. - Clean our code to be read for review ✅Fixed some bugs
Week 6	📌Form to add events to the calendar. 📌The ability to drag tasks into the calendar week 📌Make some settings options as global variable. 📌Notes folders feature 💪Improve the UX & UI design 💪improve the tasks panel

	📌Add notes panel. 📌Fix bugs
Week 7	📌Link tasks and notes to a event 📌Tasks folders feature 📌Tasks priorities 📌Search bar - the user can search for notes , tasks and events 📌Implement the improved design 📌Weather API 📌Improve timeZone 📌Fix bugs
Week 8	📌AI writing assistant 📌AI to generate Tasks 📌The ability to change events color 📌The ability to add icon to a events 📌Improve the calendar 📌Fix bugs
Week 9	📌Final touch 📌Fix bugs 📌Make a presentation (1 minute) 📌Make final report
Week 10	-

Highlights

Hugo:
Reload elements: our framework next.js loads all the required files on the server before pushing it to DOM. This means all the constant variables get set on the server first. If you have an HTML element that needs input from a variable that gets filled with data from the database, the element won't change when the data changes. For this reason, if you clicked on a note, the title and description would be loaded only once. So that means if you clicked on a new note, the value of that element would not change. We solved this by adding an event listener to the router. If the route changed, we would fire a location.reload so that the server would load in the new values and load the new element with the correct values.

Saving the notes: With our library slate, we would be able to make a rich text editor. You can only save if you push the save button; that way, you cannot read the value from the slate editor. So I made an event handler that detects if there is a change in the slate editor component and if so it sets a global variable to that value. Because in the slate component you can read the value. This way the variable can be saved in the database.

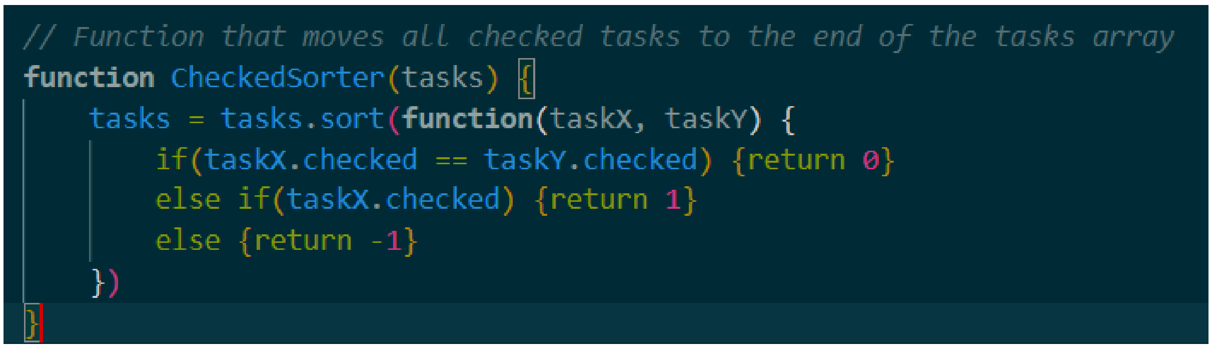
Sorting the notes: We made it simple to implement multiple sorting algorithms for the titles of the notes. By making a free copy of the current data, sorting that copy and then setting the data to that copy you will be able to sort the notes to your liking.

Elia:
Dynamic pages: are pages that are generated on-demand at runtime based on data that is not known during build time.

As an example, we are allowing the users to create notes in the note page. Each note has a unique ID, and we created a dynamic page that shows the details for a given note. In this case, we created a dynamic page that accepts a note ID as a route parameter, fetches the data for that specific note from a database, and then displays the details of that note on the page. This allows us to generate a unique page for each note that is created, without having to pre-build all of the pages during the build process. This can be useful for improving the performance of our app and making it more flexible.

Global variable: you can store data in a variable and access/change this data from different pages. I wanted to be able to show and hide the setting in every component without having to pass the state props for every component.

Pepijn:
Sorting the tasks: it took me some time to figure out how to create sorting functions for my tasks arrays. At first I was considering programming sorting algorithms from the ground up. Luckily however, I eventually came across the built-in sort method from javascript. This method sorts an array with a given function. This function is to compare two items *a* and *b* of the array and then return either a zero, positive number or negative number to the sort method. When the function returns a zero the items will remain in the same position in the array, when a negative number is returned item *a* is placed before item *b* in the array, when a positive number is returned item *b* is placed before item *a* in the array. So as an example:



```
// Function that moves all checked tasks to the end of the tasks array
function CheckedSorter(tasks) {
    tasks = tasks.sort(function(taskX, taskY) {
        if(taskX.checked == taskY.checked) {return 0}
        else if(taskX.checked) {return 1}
        else {return -1}
    })
}
```

Using this knowledge I managed to make a *AlpabeticalAscendingSorter(tasks)* function, a *AlphbaticalDescendingSorter(tasks)* function, a *DateAscendingSorter(tasks)*, a *DateDescendingSorter(tasks)* function and a *CheckedSorter(tasks)* function (as seen above).

Group Reflection

We as a group have learned how to work structured in a big project. Using Git helped a lot in this. Making branches for each page gave the ability to work separately on the same website without getting a lot of conflicts. Our web development skills have improved a lot. Next.js made it really easy for us to implement components and saved us a lot of time. The use of databases in connection with web development is also a skill we can add to our list.

We have also learned that communication and good planning are important for working on big projects. Each week we had three separate meetings. Monday we did a code review, merged our branches and coded together for a few hours, Tuesday we discussed how the sprint is going and what plans we made for this sprint and on Friday we planned the next sprint together and solved a few problems we still had for the last sprint. This way we are all up to date on what everyone is doing and how everything is going.

This is something that we need to keep up for the rest of the project. Last sprint we discussed one other thing. For each sprint, we plan 3 main tasks and we divide those main tasks into subtasks. For the next half of the project, we really need to tell the group when a subtask is done. We need to commit the changes to the branch we are working on and send a text in the group chat so that others can review the subtask and add comments on what should be improved or what is really good.