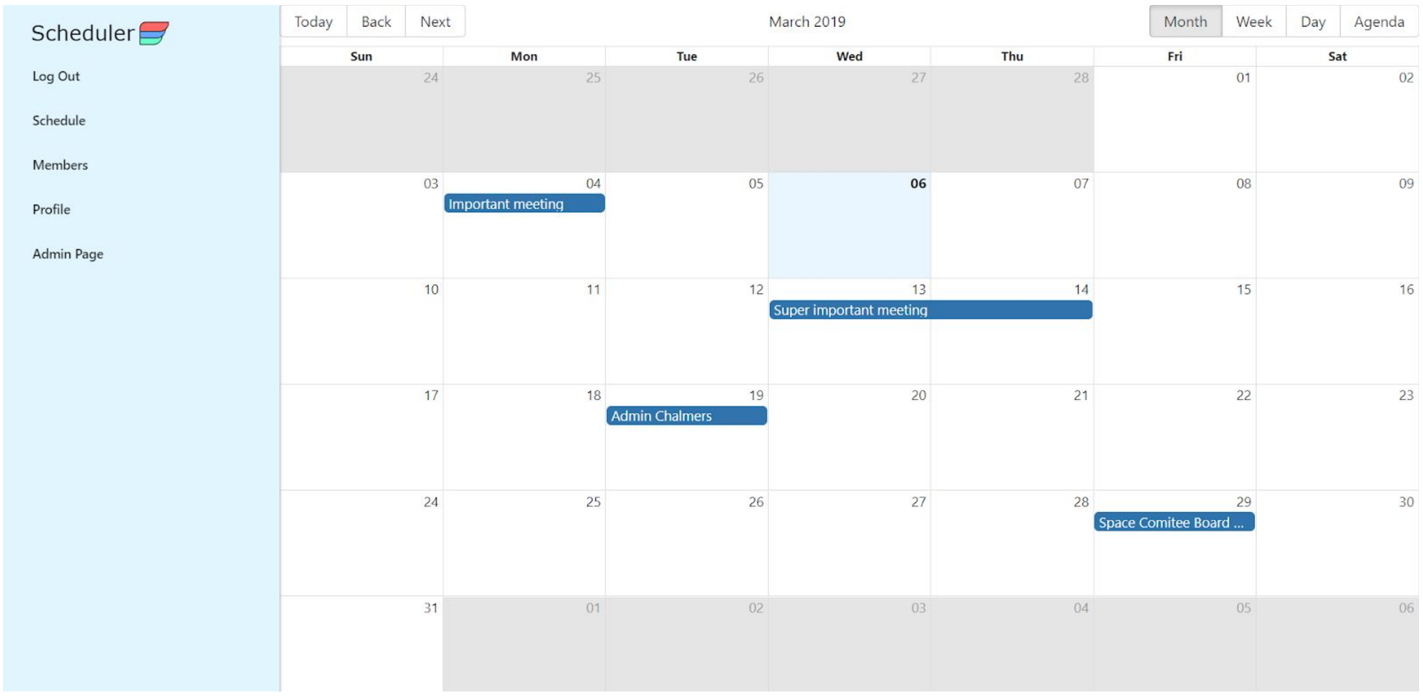
Scheduler
DAT076 - Group 14



Sun	Mon	Tue	Wed	Thu	Fri	Sat
24	25	26	27	28	01	02
03	04 Important meeting	05	06	07	08	09
10	11	12	13 Super important meeting	14	15	16
17	18	19 Admin Chalmers	20	21	22	23
24	25	26	27	28	29 Space Comitee Board ...	30
31	01	02	03	04	05	06

Project Members, Group 14:

Martin Karlsson	karlsma@student.chalmers.se	Datateknik 180 HP
Oscar Nilsson	nioscar@student.chalmers.se	Datateknik 180 HP
Anton Wester	weanton@student.chalmers.se	Datateknik 180 HP
Tobias Rosengren	tobros@student.chalmers.se	Datateknik 180 HP

Table of Contents

1. Introduction

The aim of this project was to create an online schedule application meant for business purposes such as an organization creating events or meetings. Members of the organization can then log in with their own account and view all the events present in the schedule and sign up to them.

The application was designed using Javascript, the React JS Library [1] and the state container Redux [2]. For CSS resources Materialize [3] was used. Data storage and authentication is handled on Google's Firebase platform [4].

The services used on the firebase platform specifically were the user authentication and Firestore for storing data in a NoSQL database.

The code for the project can be found in the following Git repo along with a readme describing how to set up the project from your own computer:

https://github.com/oscarnnn/DAT076-Schedule

2. Use cases

In the applications finished state, these are all the use cases based on if the user is an administrator or regular user. It should also be noted that only the application developers with direct access to the database can appoint new users as administrators. The administrators have the same use cases as regular users as well as those listed under administrator use cases.

2.1 User

- Create an account
- Log in
- Log out
- View schedule
- Participate in events
- Leave events
- View a list of members in the organization the user belongs to
- Edit user profile

2.2 Admin

- Create new events
- Edit existing events
- Delete existing events
- Add members to your organization
- Create an organization

3. User Manual

When users enter the application they have two options that are located in the side navigation bar. Either log in through the “Log In” view or create an account through the “Sign Up” view.

Once they have done either of the aforementioned choices they are then redirected to the schedule view and the navigation bar is updated with new choices. If you’re a regular user then you’ll have the option to log out, reach the schedule view where you can view event information in the schedule and sign up for the event, reach the members view where you can see the members of your organization and their contact information, and to reach your profile view where you can edit your profile.

If you are using an admin account you’ll also have access to the admin view where you can create an organization if you’re not part of one, or add members to your organization if you are already part of an organization.
As an admin you’ll also be able to add, edit and delete events from the schedule through simply clicking the schedule to add an event or click an existing event to edit or delete it.

4. Design

As mentioned previously, the application was designed in Javascript and mainly using the React JS Library which offers a variety of features and design patterns to create both functionality and intuitive graphical representations of different components.

4.1 npm - Node.js package manager

For managing packages as well as setting up the base of the project, the tool npm was used [5]. This greatly helped in speeding up the development process as installing new libraries and dependencies is made easy with npm.

4.2 React.js

The main library used for designing the application was React.js [1]. The reason being the development team had previous experience using it and it makes for a great tool when it comes to creating simple and highly functional user interfaces. The language used when working with React.js is javascript as well as JSX [6].

4.3 The Component pattern

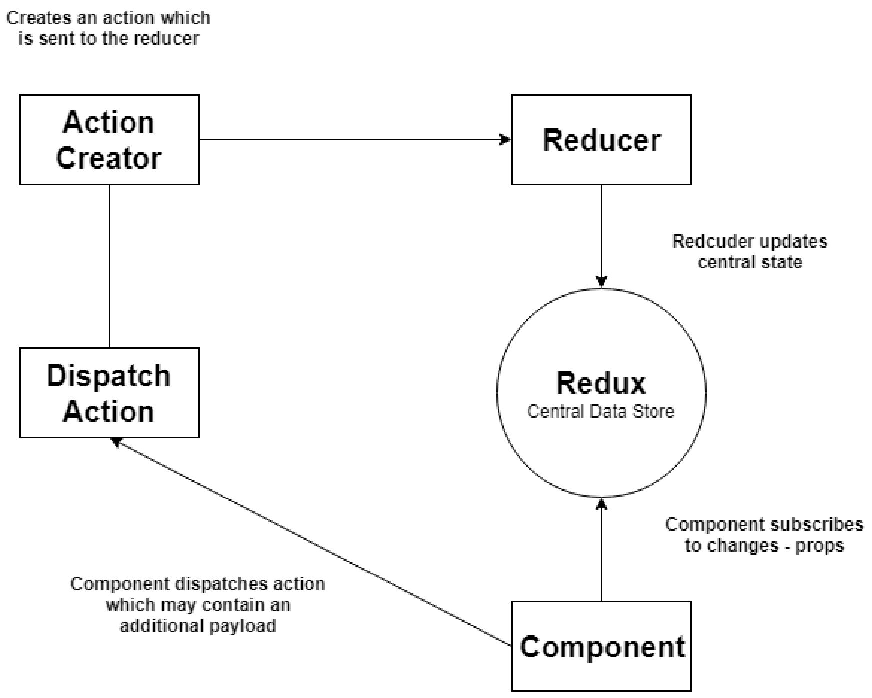
The main design pattern used when programming with the React JS library is the Component-class which is used to extend Javascript classes to easily be able to represent them graphically and to easily assign properties to the class which can then be used to alter or represent the state of the class objects.

Each Component class comes with a render function which is used to generate the HTML elements and assign functionality to the rendered components.

```
Class <ClassName> extends Component {
      render() {
            return (<content to be rendered>)
      }
}
```

4.4 Redux and Firestore

Each component with a need for database access contains a function which connects the component to the applications central state. When this state changes the components state is automatically updated. The database operations are performed in the actions which are then dispatched by the component. The changes to the database are then reflected in the central state through the components corresponding reducer.


Creates an action which is sent to the reducer
Action Creator
Reducer
Redcuder updates central state
Dispatch Action
Redux
Central Data Store
Component subscribes to changes - props
Component dispatches action which may contain an additional payload
Component

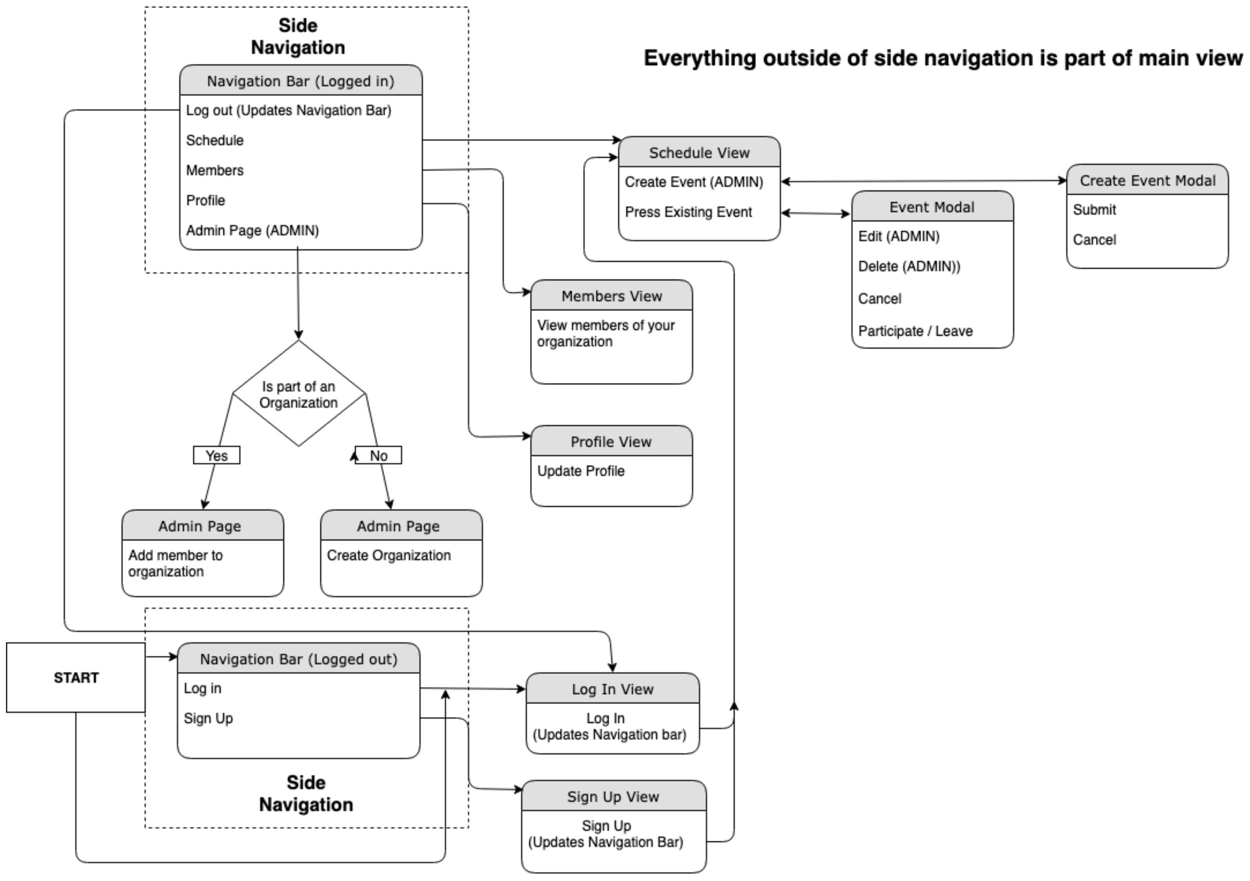
4.5 Libraries

In addition to React JS and Redux some additional libraries were used. These are listed below along with their roles:

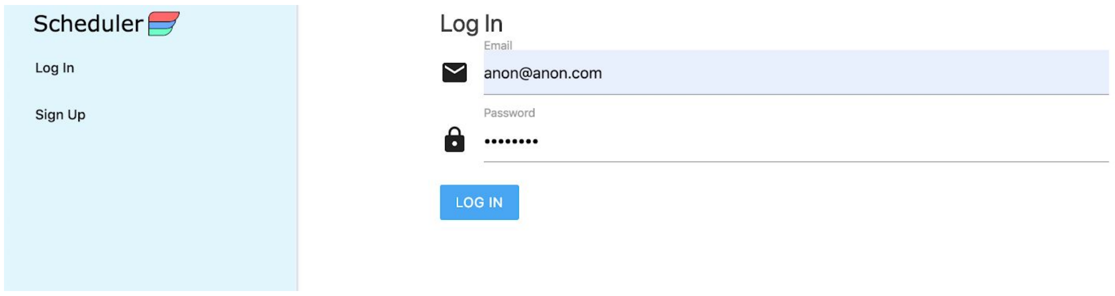
● react-big-calendar	- Creating and displaying the calendar and events
● react-date-picker	- Pick start and end dates when creating/editing events
● react-router-dom	- Used for routing in the application
● Materialize	- Used for CSS and styling
● redux-thunk	- Thunk middleware for Redux
● redux-firestore	- Redux bindings for Firestore
● react-redux	- Official React bindings for Redux
● react-redux-firebase	- Redux bindings for Firebase

5. Application Flow

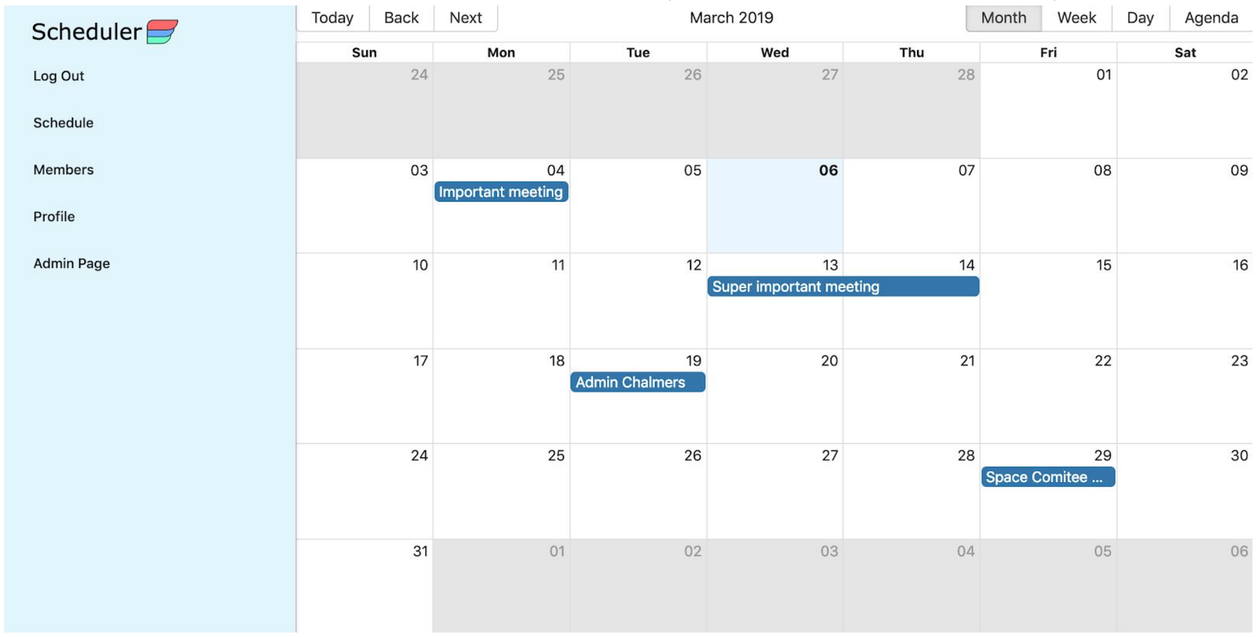
A simple flowchart that explains the application flow described earlier in section 3.



The user starts with the “Navigation Bar (Logged out)” in the “Log In” view.



Once users log in they reach the “Schedule” view and the navigation is updated.
From here the user can access different functionality based on the user’s authority level.



Sun	Mon	Tue	Wed	Thu	Fri	Sat
24	25	26	27	28	01	02
03	04 Important meeting	05	06	07	08	09
10	11	12	13 Super important meeting	14	15	16
17	18	19 Admin Chalmers	20	21	22	23
24	25	26	27	28	29 Space Comitee ...	30
31	01	02	03	04	05	06

6. Responsibilities

Martins main responsibilities revolved around the authentication, design and navigation.
He integrated the authentication from firebase in the sign in and sign up view that is present in the whole application to make sure that the users are authorized to view the content they are trying to reach. Designwise he developed the application logo and took part in a lot of the visualizing of the application.
He also developed our current iteration of the navigation system in the application through utilizing the aforementioned responsive CSS framework called Materialize.

Antons primary responsibilities revolved around the views for members, your profile and parts of the admin view. He mainly took part in developing the profile view were you can view and edit profile information connected to the Firestore database, the member view were you can view every member in your organization and their contact details, and the part of the admin view that is shown when an admin is not part of an organization and can then create an organization through that view.

Oscars main responsibilities revolved around everything that had to do with the schedule and events of the application which was the most vital view of the application, he also took part in a lot of refactoring that was done during the project. He implemented the calendar and almost everything related to it which is a vital component of the application.

Tobias primary responsibilities revolved around the admin, member and schedule views.
He, together with Oscar, handled most matters related to the schedule and visualizing events in the calendar, which is one of the most vital views of the application. Tobias also took part in visualizing the user collection from Firestore in the member view. Other than that he and Oscar also developed the admin functionality of adding members to your organization through the admin view. He also designed some of the actions and reducers for the aforementioned parts of the project.

7. References

[1] Reactjs.org. (2019). *React – A JavaScript library for building user interfaces*. [online] Available at: https://reactjs.org/ [Accessed 6 Mar. 2019].
[2] Redux.js.org. (2019). *Redux · A Predictable State Container for JS Apps*. [online] Available at: https://redux.js.org/ [Accessed 6 Mar. 2019].
[3] Materializecss.com. (2019). *Documentation - Materialize*. [online] Available at: https://materializecss.com/ [Accessed 6 Mar. 2019].
[4] Firebase. (2019). *Firebase*. [online] Available at: https://firebase.google.com/ [Accessed 6 Mar. 2019].
[5] Npmjs.com. (2019). npm. [online] Available at: https://www.npmjs.com/ [Accessed 15 Mar. 2019].
[6] Npmjs.com. (2019). *npm*. [online] Available at: https://www.npmjs.com/ [Accessed 15 Mar. 2019].