



# T-404-LOKA, Final Project

## Tempo Chrome Extension

### **Final Report**

Anton Marinó Stefánsson

Hrönn Róbertsdóttir

Sigrún Þorsteinsdóttir

Þórdís Jóna Jónsdóttir

Spring 2016

Instructor: Haukur Kristinsson

Examiner: Elín Elísabet Torfadóttir



# Abstract

The aim of this project is to create an extension for the popular browser Chrome, that allows its users to easily track time and log their work. The extension will integrate seamlessly with Tempo Timesheets for JIRA and it will also allow users an overview of their Google Calendar events. The hope is that Tempo will be able to present the extension to their clients as an addition to the JIRA solutions they already offer - that help teams collaborate, plan, track, and achieve their goals.

The aforementioned project constitutes as a partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

# Review

It has been a pleasure working on the Tempo for Chrome project with Anton Stefánsson, Hrönn Róbertsdóttir, Sigrún Þorsteinsdóttir and Þórdís Jónsdóttir. The team has shown that they can work on a new project and solve problems that arise during development quickly and efficiently as well as keeping an eye on the user experience needed throughout the product development process.

Their work ethic has been excellent where they have delegated tasks between team members. Each member has shown responsibility with their tasks and communicating with the team on their progress.

---

*Elías R. Ragnarsson*

*Product Owner*

*Tempo*

# Table of contents

Introduction	1
<b>1. Methodologies and tools</b>	2-4
1.1 Scrum	2-3
Roles	2-3
1.2 Work Conditions	3
1.3 Team Productivity	3-4
<b>2. Design and development</b>	5-12
2.1 Programming Rules	5
2.2 Test design	5-6
2.3 Software architecture	6-9
System design	6
React	7
JSX	7
Redux	7-8
ES6	8
Node.js	8
APIs	8
SuperAgent	9
Bundling tools	9
Webpack	9
SCSS	9
2.4 Functionality	10-12
<b>3. Project progress</b>	13-21
3.1 Sprints	13
3.2 Progress Overview	14-20
Overview of total work hours	14-15
Sprint 1 (24.jan - 13.feb)	15
Sprint 2 (14.feb - 5.mar)	16
Sprint 3 (6.mar - 26.mar)	17
Sprint 4 (27.mar - 16.apr)	17-18
Sprint 5 (17.apr - 23.apr)	18
Sprint 6 (24.apr - 30.apr)	19
Sprint 7 (1.maí - 7.maí)	19
Sprint 8 (8.maí - 13.maí)	20
3.3 Project Burndown Chart	21
<b>4. Risks</b>	22-23
4.1 Risk analysis table	22
<b>5. Future work</b>	23
<b>6. Conclusion</b>	24
Bibliography	25

# Introduction

The team had the privilege of doing a project for Tempo, a company that specializes in mission critical solutions designed to enhance the efficiency of JIRA and help software, professional, and small business teams collaborate, plan, track and achieve their goals. [1]

Tempo is a subsidiary of Nýherji that was founded within a different subsidiary of Nýherji, TM Software. The initial idea of Tempo emerged at a hackathon within TM Software, which then lead to a startup company. Today, this once small startup employs more than 70 people who are working on developing four different products for multiple companies. Tempo is working on developing their own take on JIRA's time management solutions.

The main focus of Tempo's products is to make it easier for people to manage their work hours and plan their workdays. This is precisely what one of their products, Tempo Timesheets, is all about - tracking work hours and overseeing one's daily work agenda.

The idea for this project came from multiple user requests that Tempo received asking them to build a Chrome extension for their Tempo Timesheets product. They wanted Tempo to ease the process of tracking time and viewing one's work schedule. The extension will enable users easy access to their JIRA and Google Calendar accounts inside the Google Chrome browser when opening the extension which is always accessible in the browser's tool bar.

The main focus of the extension was to make it easy to use so that users could manage their work hours and log time in as few steps as possible.

This report contains the process of implementing the extension. In chapter one the Scrum methodology will be explained amongst other tools used to manage the project. In chapter two the whole technology stack will be explained along with the functionality that the extension provides. In chapter three an overview of the project's progress will be given along with graphs and descriptions for each sprint. In chapter four the project's risk factors will be reviewed. In chapter five a short overview will be given of possible future work concerning the extension. In chapter six the report will be concluded with thoughts and the final outcome of the project.

# 1. Methodologies and tools

This chapter contains information on the team's use of the Scrum methodology, work conditions and productivity.

## 1.1 Scrum

The team decided to make use of Scrum - an agile software development methodology. This included iterative sprints and daily meetings at 10 am to keep track of everyone's work and make sure that everything stayed on schedule. Since one team member was working from home the daily meetings were held online with the help of Trello [2], a web-based project management application.

The team also made use of a few other components of Scrum such as user stories, a product backlog, sprint goals, sprint backlogs, burndown charts and retrospective meetings.

The product backlog is made up of user stories that are broken up into subtasks and the amount of work for each task is then estimated. Each sprint starts off with a team meeting where the sprint goal is decided. A sprint goal shows what stories to focus on during the sprint. During the course of each sprint the scrum backlog and burndown chart give a good overview of what tasks are due and whether or not the sprint goals will be met. Finally, at the end of each sprint there is a retrospective meeting with the Product owner, so that he can keep track of the project's progress.

## Roles

*Scrum Master* - Sigrún Þorsteinsdóttir

The scrum master's role is to remove all internal and external impediments that can get in the way of the team's productivity to help them reach their goal for each sprint. Other responsibilities include planning team meetings, promoting that the team maintain the scrum principles, and keeping track of work hours.

*Product Owner* - Elías R. Ragnarsson (Tempo)

The product owner, or PO, is a contact between the team and the stakeholder. He should have a vision of how he wants the final product to look and he needs to be able to convey that vision to the team. He is responsible for there being a product backlog and making sure it is prioritized.

*The Team* - Anton Marinó Stefánsson, Hrönn Róbertsdóttir and Þórdís Jóna Jónsdóttir

The scrum team is at the heart of each project. The team is a collection of individuals that work closely together to deliver what is requested by the stakeholder. If a team is to be effective it is important that they share a common goal and everyone agrees to adhere to a certain set of rules and norms.

## **1.2 Work Conditions**

Tempo provided the team with a good working facility equipped with a desk, chairs, computer screens, keyboards and mice. This facility was located at the offices of Tempo amongst their other employees so asking for guidance or assistance did not prove difficult. The Product owner and main contact within Tempo was Elías R. Ragnarsson who works as a designer for Tempo. Elías had a clear vision of what the stakeholder was expecting from the completion of the project and he was of great assistance when it came to UI design, scrum planning and review.

## **1.3 Team Productivity**

The team decided to start off strong and try to keep a steady pace of work hours during the course of the project. Each team member was to work 22 hours per week. With a team of four that equaled a total of 88 work hours per week. This was the case for most of the sprints except for the occasional weeks where team members had other big projects due. The sprints that included those weeks had a reduced estimate of total work hours. This was compensated for by adding an additional 3 work hours per week during the last four weeks of the project.

Table 1.3.1 - Total Work Hours

Project total			
Name	Estimated hours worked	Actual hours worked	Total
Anton	320	397	124%
Hrönn	320	363	113%
Sigrún	320	359	112%
Pórdís	320	378	118%
<b>Total</b>	<b>1280</b>	<b>1497</b>	<b>117%</b>

*An overview of total time spent on the project.*

In order to keep detailed track of hours worked the team made use of Toggl [3], an online time tracking tool. Each team member logged time for each day worked along with a short description on what issue was worked on that day. Toggl would then collect information for each member into a pie chart showing how many hours each member worked for any given time period.

The project backlog was stored on a board on Trello and the team would then pick stories from that backlog and move them to the sprint backlog during sprint planning meetings. That way, the team had a clear vision of what stories should be covered each sprint as each team member was assigned to a story they were in charge of.

When a story was being worked on, it was moved from the sprint backlog to the “*Doing*” column and it would stay there until it was ready to be reviewed. The team reviewed the stories together and if all members agreed that the functionalities were sufficiently implemented, the story was then moved to the “*Done*” column.

The combination of Trello and Toggl made it easy for everyone to keep track of hours worked and what tasks they were conducting each day.

## 2. Design and development

This chapter contains information on the technology stack that was used to build the product along with a short manual on the Chrome extension's main functionalities.

### 2.1 Programming Rules

In order to keep the code to a certain standard, everyone agreed to abide by a set of programming rules that were decided by all team members.

The decision was mutual to follow the Airbnb style guides [4] for JavaScript and JSX. The team used the code editor Atom and installed the Airbnb ESLint package which made sure that all group members followed the aforementioned rules.

### 2.2 Test design

Karma was picked to run the unit tests and to manage the code coverage along with Jasmine for the tests syntax. PhantomJS was then used to simulate a browser and its functionalities. Every reducer and action in the project were then tested.

The tests were designed to be straightforward, only testing one function at a time. The actions were tested by creating mock actions and mock data, passing the mock data into the actual action and then comparing the mock action to the actual action. The reducers were tested by creating mock actions, mock data and a mock state. The mock state and mock actions were then passed on to the actual reducer and finally the mock data was compared to the data that the reducer returned.

Below are two different examples of action tests. The first one is updating a value and the second one is calling the API and then updating values. The mock data was not included in the image because the team deemed it irrelevant.

```

1  import moment from 'moment';
2
3  import * as actions from '../app/actions/googleEvents.js';
4  import * as googleAPI from '../app/API/googleCalendarAPI.js';
5
6  describe('actions - googleEvents - ', () => {
7    it('should update Google events', () => {
8      const events = mockGoogleEvent;
9      const expectedAction = {
10        type: 'UPDATE_GOOGLE_EVENTS',
11        events
12      };
13      expect(actions.updateEvents(events)).toEqual(expectedAction);
14    });
15  });
16
17  describe('actions - googleEvents - async functions -', () => {
18    beforeEach(() => {
19      spyOn(googleAPI, 'getGoogleEvents');
20
21      actions.getEvents(mockGoogleGetData.token, mockGoogleGetData.calendarID, mockGoogleGetData.dates);
22    });
23
24    it('should call getGoogleEvents', () => {
25      expect(googleAPI.getGoogleEvents).toHaveBeenCalled();
26    });
27
28    it('should call getGoogleEvents with right parameters', () => {
29      const token = mockGoogleGetData.token;
30      const calendarID = mockGoogleGetData.calendarID;
31      const dates = mockGoogleGetData.dates;
32
33      expect(googleAPI.getGoogleEvents).toHaveBeenCalledWith(token, calendarID, dates);
34    });
35  });

```

*Image 2.2.1 - Example tests for actions*

## 2.3 Software architecture

This section describes the architectural decisions, and will be divided into two categories; system design and tools that help with building and bundling the solution.

### System design

Below is a list of tools that were used to implement the functionality and architecture of the extension. The extension is written in Javascript, using a library called React. Redux was picked to send data between components and make sure that the state of the action changes. Using JSX made it possible to use XML syntax in Javascript code and Babel enabled the use of the newest type of ECMAScript, without having to worry about browser support. For the backend, API calls were used to fetch all data displayed in the extension

## React

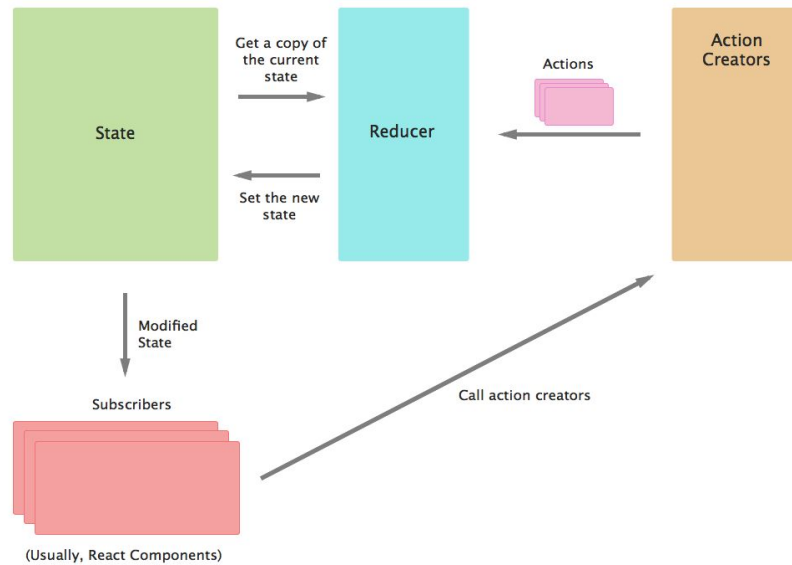
React is a popular JavaScript library for building user interfaces. It was made for building large applications with data that changes over time. React builds encapsulated components, which makes code reuse, testing and separations of concerns easy. React also only reloads the part of the DOM that was modified resulting in better performance, so it handles data binding very well. [5]

## JSX

JSX is a JavaScript syntax extension that allows the user to add XML syntax to JavaScript. It is recommended to use JSX instead of regular JS with React because it is a concise and familiar syntax for defining tree structures with attributes. JSX also shows compile-time errors that, for example, notify the user when the code does not compile because of spelling mistakes. [6]

## Redux

For the projects architecture the team used Redux. Redux uses unidirectional data flow that uses a single store to hold data. This store is changed by cloning the original store and applying functions without side effects. Redux was picked to simplify the applications architecture. The store contains the state for the entire application and is organized in a tree of objects. So when a state has to be changed, a copy is made of the previous tree and a new tree is added with the changed state. The store has only one source of information which is called an action. Actions are payloads of information that send data from the application to the store. Actions only describe the fact that something happened, but they do not specify how the application's state changes in response. That is the job of a reducer. The reducer is a pure function that takes the previous state and an action and returns the next state. [7]



*Image 2.3.1 - Overview of the flow of the Redux process. [8]*

## ES6

The team used Babel to translate the newest type of ECMAScript to a code that the browser can understand. The programmer can thus write code using the newest ECMAScript syntax and let Babel take care of compiling the code to the ECMAScript version that the user's browser understands. [9]

## Node.js

The team decided to use Node.js for the npm package manager. Npm made the process to install dependencies for the project easy and manageable.

## APIs

The project included four different APIs to gather data; Tempo Timesheets REST API, Google Calendar API, JIRA API and Superagent API.

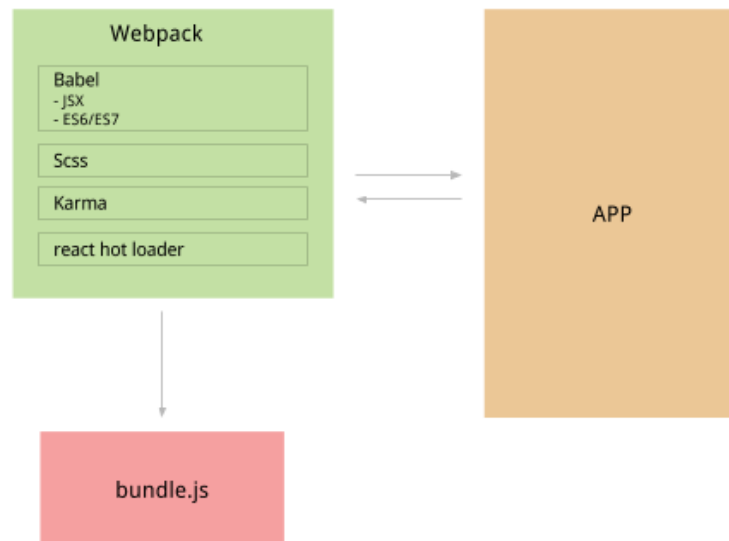
## SuperAgent

SuperAgent is a lightweight progressive AJAX API and was picked because of its flexibility, readability and low learning curve. For this project, Superagent was used to send requests to APIs. [10]

## Bundling tools

### Webpack

Webpack is a module bundler which takes modules with dependencies and emits static assets representing those modules. This is used to reduce the initial loading time of the extension. Webpack splits the codebase into multiple chunks and can be loaded on demand. Webpack was chosen over tools like Browserify and Gulp. The main reason for that was because of its capability to work well with the aforementioned tools that had already been chosen. [11]



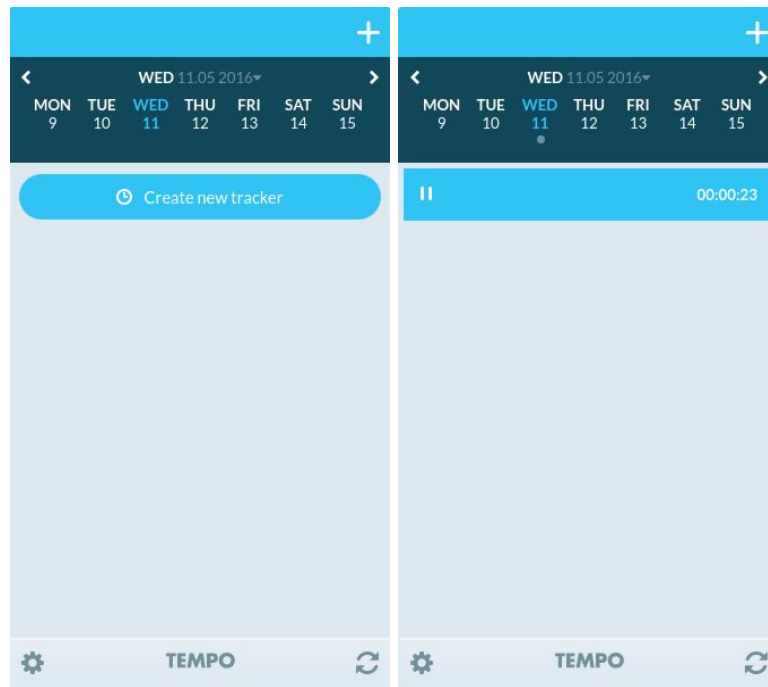
*Image 2.3.2 - Overview of the tools that Webpack uses to bundle dependencies together.*

### SCSS

SCSS is an extension to the old CSS styling language. SCSS is a superset of the CSS3 syntax but allows more advanced usage such as variables and extending already defined code. Using Webpack, the SCSS code will then be translated to regular CSS code on runtime.

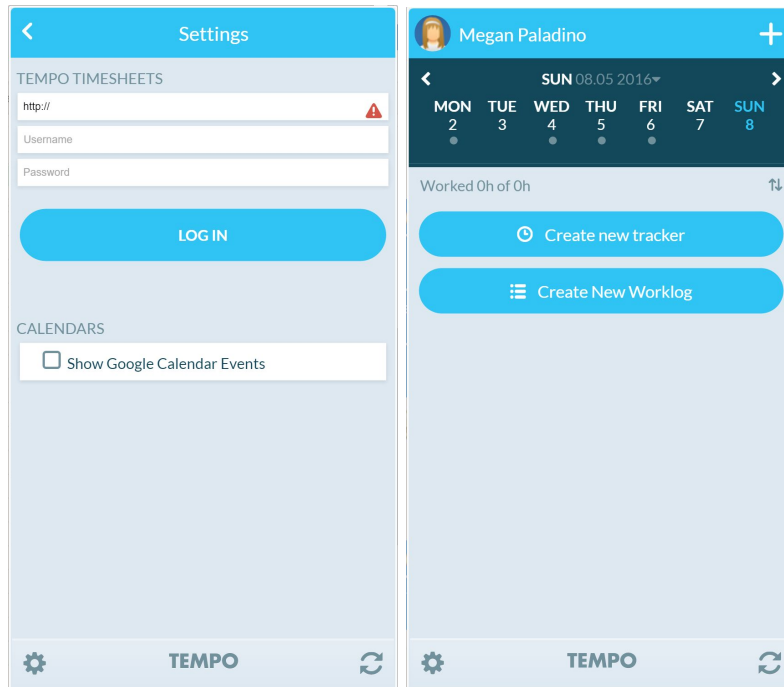
## 2.4 Functionality

This section describes what functionalities the Chrome extension offers. Below are six different views that show the main features of the extension. For detailed usage, please refer to the user manual (*Appendix A*).



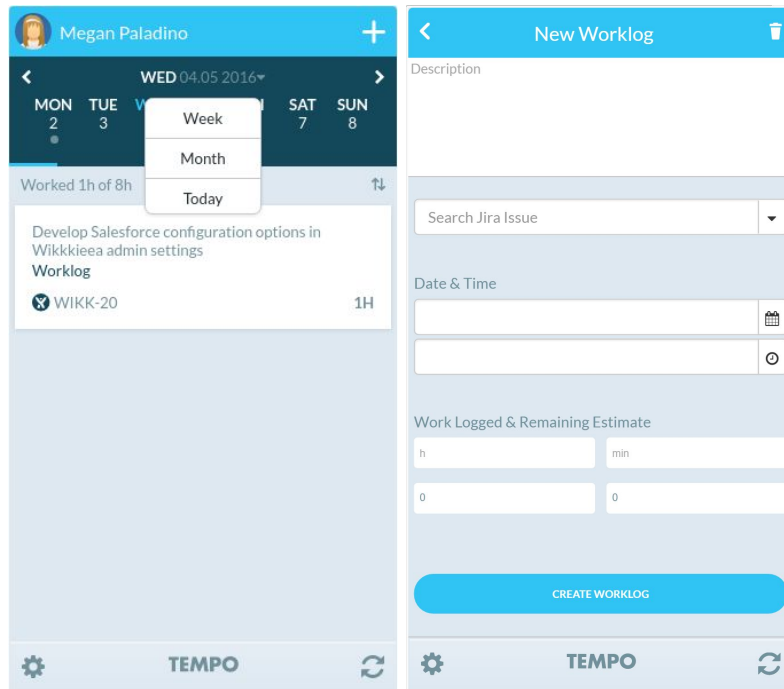
*Image 2.4.1 - Initial view for unidentified users (left) and tracker created by unidentified user (right)*

This is the initial view for a user that is not signed in to a JIRA account nor a Google account. The user is still able to create a new tracker by pressing the “*Create new tracker*” button. The user can also navigate through the calendar.



*Image 2.4.2 - Settings view (left) and main view for identified user (right)*

The user can press the cogwheel on the bottom left corner that will lead him to the view which the left part of *Image 2.4.2* shows. The user can then log into his JIRA or Google account. The right part of *Image 2.4.2* shows the main view of an identified user that can now additionally view worklogs and Google events on his calendar. He can also choose to create, edit or delete a worklog and convert a Google event to a worklog. The gray dots beneath each day on the calendar indicate that there is a Google event, a worklog or a tracker assigned to the user that day.



*Image 2.4.3 - Initial view for identified user with a worklog showing (left) and “Create new worklog” view (right)*

The left part of *Image 2.4.3* shows the user’s worklog for the day, and the user has clicked the dropdown arrow on the calendar and can now change between week- and month-view or navigate back to today. The right part of *Image 2.4.3* shows the “Create new worklog” view where the user can write a description, select an issue to assign to the worklog, pick a desired time for the worklog and finally create it.

## 3. Project progress

This chapter will cover the project progress where each sprint is reviewed and a sprint burndown chart is shown for each sprint along with a burndown chart for the whole project.

### 3.1 Sprints

There were a total of eight sprints, not including sprint zero which was a week long preparation sprint. The first four sprints were three weeks each followed by four sprints that were one week each as can be seen in *Table 3.1.1*.

*Table 3.1.1 - Sprint Retrospective Table*

Sprint nr.	Date	Retrospective
Sprint 1	24.jan - 13.feb	Start working on developer stories from product backlog - doing research and building boilerplate. Continue working on the product backlog.
Sprint 2	14.feb - 5.mar	Start working on user stories from product backlog - connecting the Chrome extension to JIRA API and setting up the calendar. Continue working on finishing up the boilerplate.
Sprint 3	6.mar - 26.mar	Start working on connecting the Chrome extension to Google API, searching JIRA issues, unit testing the calendar and the overall UI. Continue working on the calendar.
Sprint 4	27.mar - 16.apr	Start working on creating worklogs and the card component. Stop unit testing. Continue working on the calendar and UI.
Sprint 5	17.apr - 23.apr	Start establishing a connection to the Google API (again). Continue working on the card component and the UI.
Sprint 6	24.apr - 30.apr	Start working on the tracker component. Continue working on the overall look.
Sprint 7	1.may - 7.may	Start getting everything ready for code freeze. Continue doing unit tests, and working on the UI and the final report.
Sprint 8	8.may - 13.may*	Stop adding new functions (code freeze). Continue doing unit tests and working on the final report.
<b>Final product</b>	<b>13.may</b>	

\*The eighth sprint is a day short of a week due to the hand-in date of the final product

## 3.2 Progress Overview

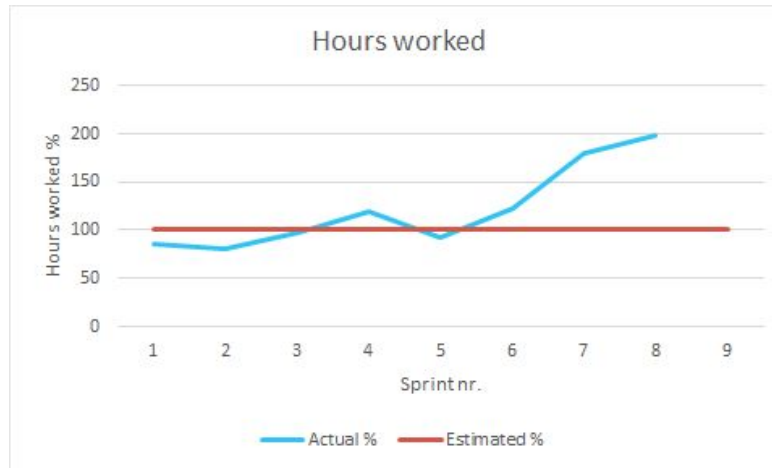
Below is a table and a chart showing how many hours were spent on each sprint along with burndown charts and descriptions for each sprint.

### Overview of total work hours

*Table 3.2.1 - Sprint Work Hours*

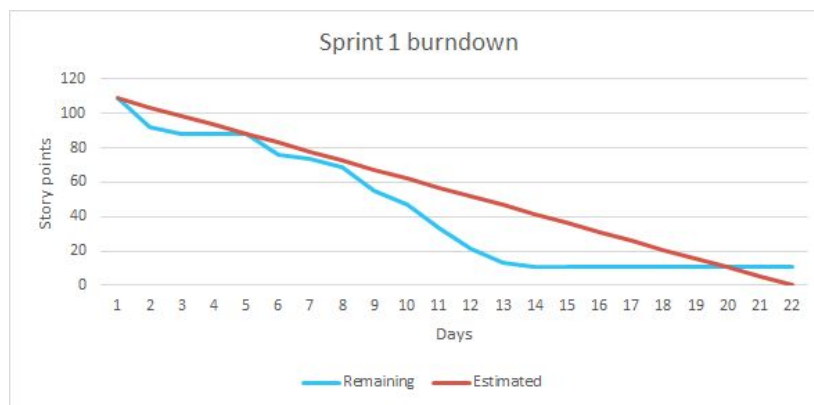
<b>Sprint nr.</b>	<b>Estimated hours worked</b>	<b>Actual hours worked</b>	<b>%</b>
0	88	75	85
1	264	215	81
2	176	172	98
3	176	209	119
4	176	163	93
5	100	123	123
6	100	186	186
7	100	247	247
8	100	107	107
<b>Total</b>	<b>1280</b>	<b>1497</b>	<b>117</b>

*An overview of hours worked per sprint.*



*Image 3.2.1 - Shows the percentage of hours worked for each sprint compared to 100%.*

## Sprint 1 (24.jan - 13.feb)

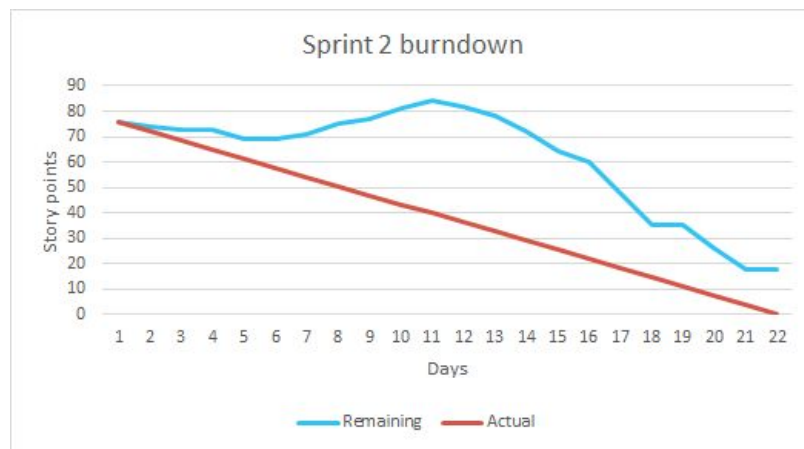


*Image 3.2.2 - Burndown chart for first sprint.*

After the preparation sprint the goal was to take on the developer user stories from the product backlog. This meant a lot of time spent researching and studying up on React, Redux, and every other tool concerning the project.

It was Tempo that suggested using React and the team was excited to take on the challenge. However, no team member had worked with React before so it took a while to get familiar with it. In the first sprint a boilerplate for the project was set up where all the tools were connected together before the coding could begin.

## Sprint 2 (14.feb - 5.mar)



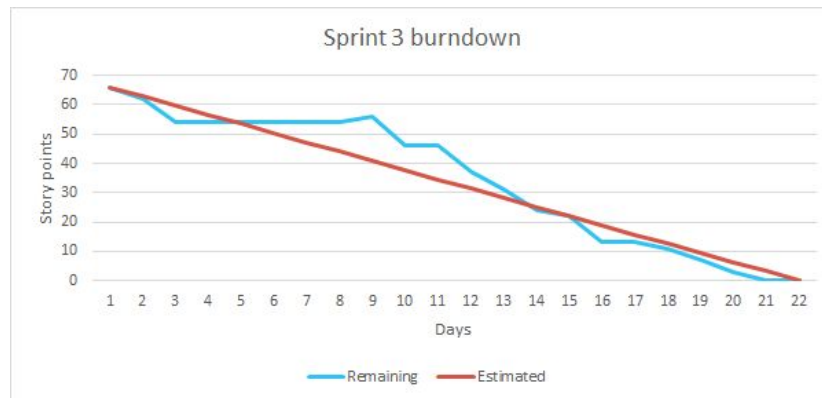
*Image 3.2.3 - Burndown chart for second sprint.*

After finishing the developer stories, the next step was a jump into the deep end with setting up a connection with the JIRA API. This took quite a while but after a pair programming session, a connection to JIRA API was finally established.

It was then that the team realized that the time needed to complete said tasks had been wildly underestimated so it was decided to review the product backlog and re-evaluate the amount of story points set for each story.

During this sprint, work on the Calendar component also began and it got off to a flying start.

### Sprint 3 (6.mar - 26.mar)



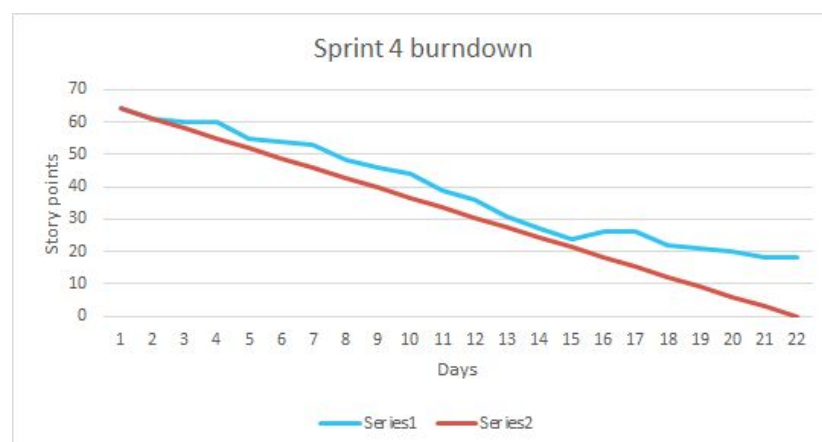
*Image 3.2.4 - Burndown chart for third sprint.*

Once the connection to the JIRA API had been established, all data needed for the worklogs could now be fetched.

Thus, the third sprint focused on making it possible to search JIRA issues and create new worklogs and finishing up the Calendar component.

A connection to the Google Calendar API was needed so that process was also started.

### Sprint 4 (27.mar - 16.apr)



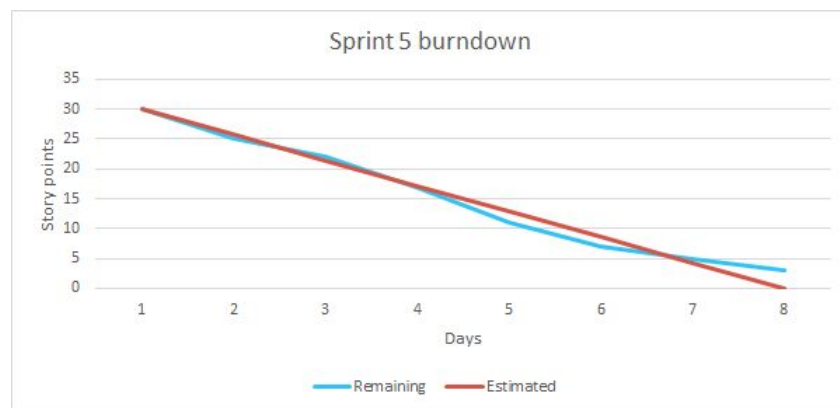
*Image 3.2.5 - Burndown chart for fourth sprint.*

When the fourth sprint came around the team had found it's footing, and everyone was feeling a lot more confident in their abilities. Everything had started to click and the ball finally started rolling.

Most of the functionality was either ready or in progress so it was decided to start diverting more manpower into working on the UI alongside the functions.

Once the UI had been added the extension finally started to look like a complete product.

### Sprint 5 (17.apr - 23.apr)

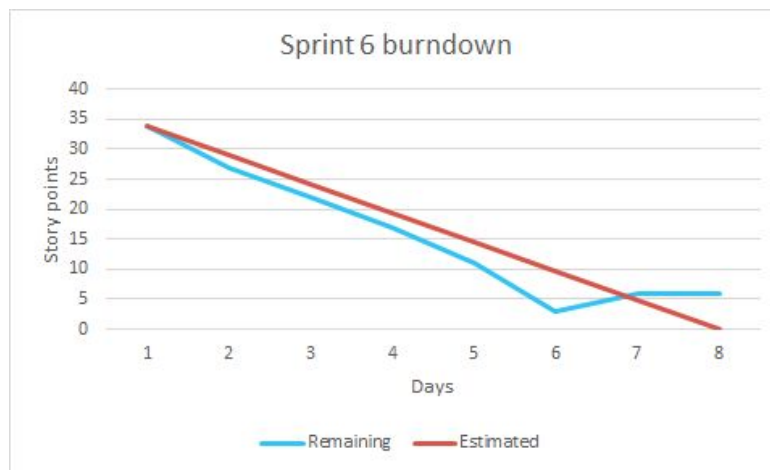


*Image 3.2.6 - Burndown chart for fifth sprint.*

The fifth sprint marked a change in pace as it was the first week long sprint but the team managed to stay on schedule. Everyone had their tasks laid out so the main focus could be on releasing a working product.

After taking a closer look at the Google API it became clear that Extensions for Chrome doesn't support the Google API that the team had picked out. After coming to a mutual decision with Tempo, the Google login was adjusted to the extension.

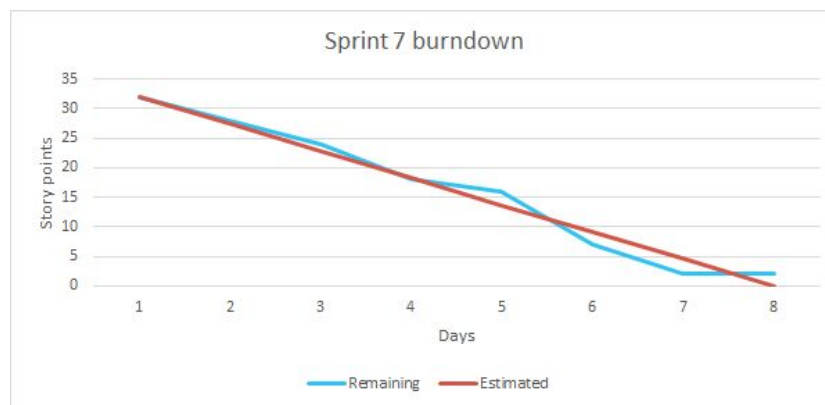
## Sprint 6 (24.apr - 30.apr)



*Image 3.2.7 - Burndown chart for sixth sprint.*

With everything else well under way the only thing missing was the Tracker component so getting that ready was at the top of the list.

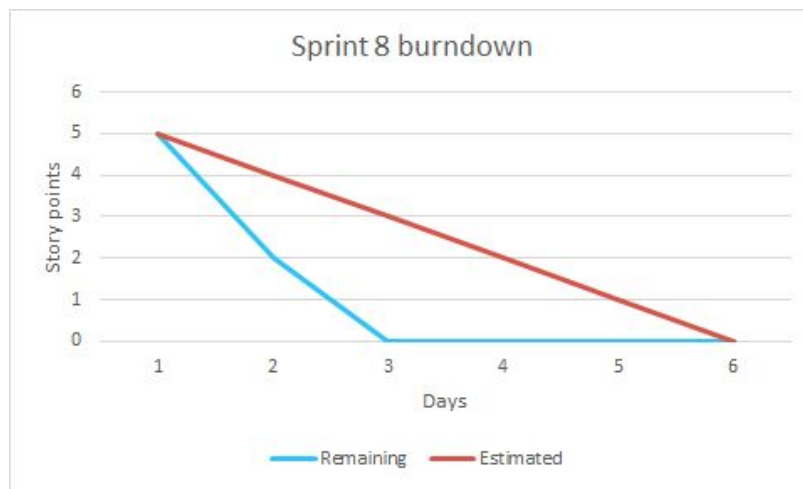
## Sprint 7 (1.mai - 7.mai)



*Image 3.2.8 - Burndown chart for seventh sprint.*

The seventh sprint focused on unit testing, writing the final report, and polishing everything up and getting it ready for the final status meeting.

## Sprint 8 (8.mai - 13.mai)



*Image 3.2.9 - Burndown chart for eight sprint.*

There was only a handful of story points left when it came to the final sprint of the project, and they were quickly completed so that the team could focus on continuing with the unit testing and finishing up the final report.

### 3.3 Project Burndown Chart



*Image 3.3.1 - Burndown chart for entire project.*

The team managed to complete all of the A requirements from the projects product backlog. However, there were a few B requirements left unfinished that would make a nice addition to the final product.

## 4. Risks

This chapter covers any risks that may have been encountered during the development of the extension.

### 4.1 Risk analysis table

The risk analysis table lists the risks concerning the development process. The likeliness of a risk was based on a scale from one to five, five being the most likely. An identical scale was used for the impact of a risk, with risks with a five having the biggest impact. These two numbers were then multiplied to calculate the risk as a whole.

*Table 4.1.1 - Risk Analysis Table*

Risk factor nr.	Risk factor	Likelihood	Impact	Risk	Guarantor	Reduce risk	Risk management approach	Resolved date
1	Working with Redux for the first time	3	5	15	Everyone	Research Redux and learn as much as possible about it. Follow documentation as much as possible.	Team members will help each other, if that doesn't suffice we will look for outside help.	28.feb
2	Creating a Chrome extension for the first time	2	4	8	Everyone	Take a look at other Chrome extensions to get a better feel for how it's done.	Team members will help each other, if that doesn't suffice we will look for outside help.	16.feb
3	Writing in JSX for the first time	2	3	6	Everyone	Research JSX and learn as much as possible about it. Follow documentation as best as we can.	Team members will help each other, if that doesn't suffice we will look for outside help.	16.feb
4	Testing before coding, nobody has experience doing this	2	2	4	Everyone	Make sure everyone writes tests and then make sure it fails before writing code.	Team members will help each other, if that doesn't suffice we will look for outside help.	Unresolved

*Risk analysis table including resolve dates.*

Unfortunately the team was unable to set a resolve date for the last risk, seeing as the plan to put testing before coding failed. There were a few things that prevented this from happening, one of which was the decision of a CSS syntax that worked well with Mocha, our original testing framework. However, after setting up a Karma config to keep track of code coverage, it was decided it would be best to switch from Mocha to Jasmine.

## 5. Future work

The future of the solution is unclear as of now, although it bears mentioning that the Product owner has expressed interest in expanding the solution for other browsers and as an Apple widget. However, there is a short list of requirements that the team would have liked to have implemented, at the top of which is keyboard shortcuts and advanced style features.

Unfortunately time ran out so those features will have to wait for a later date.

## 6. Conclusion

At the beginning of the project, Tempo indicated an interest in developing the extension in React which was new territory for the team. Nevertheless it was a unanimous decision to take on the challenge and welcome the experience. It got off to a slow start with everyone spending most of their time researching and reading up on all the tools needed to manage the application.

Around the middle of the semester things were in full swing and most of the functions were either complete or already underway, so it was decided to start divvying up the UI workload and start unit testing. While the UI work got off to a great start, the same couldn't be said about the testing. The initial idea was to use PostCSS for the UI and Mocha for testing, but soon problems started popping up. It seemed PostCSS and Mocha didn't really work well together so the decision was made to switch over to SCSS. Later on the Mocha test framework was also switched out for Jasmine which works well with the test runner Karma, which included code coverage.

This project was a big challenge for the whole team, but even though there were a few problems along the way, for instance, regarding the Google Calendar API and login, the project was a smashing success.

Of course there are things that could have been done better, like including the testing and reports in the product backlog, but this was a great learning experience and everyone feels better prepared for what the future as a computer scientist holds.

# Bibliography

1. Tempo. (2016). *Tempo*. Retrieved 2 May, 2016, from <http://tempo.io/>
2. Trello. (2016). *Trello*. Retrieved 12 May, 2016, from <https://trello.com/>
3. Toggl. (2016). *Toggl*. Retrieved 12 May, 2016, from <https://toggl.com/>
4. Airbnb. (2016). *Airbnb*. Retrieved 12 May, 2016, from <https://github.com/airbnb/javascript>
5. Github. (2016). *React*. Retrieved 4 May, 2016, from <https://facebook.github.io/react/>
6. Github. (2016). *JSX*. Retrieved 4 May, 2016, from <https://facebook.github.io/react/docs/jsx-in-depth.html>
7. Redux. (2016). *Redux*. Retrieved 5 May, 2016, from <http://redux.js.org>
8. Kadir. (2015). *Rethinking Redux*. Retrieved 12 May, 2016 from <https://voice.kadir.io/rethinking-redux-f1e96daba60c#.ienl45q1f>
9. Babel JS. (2016). *Babeljs*. Retrieved 5 May, 2016, from <https://babeljs.io/>
10. Github. (2016). *Github*. Retrieved 9 May, 2016, from <https://visionmedia.github.io/superagent/>
11. Github. (2016). *Webpack*. Retrieved 9 May, 2016, from <http://webpack.github.io/>

# 13 May, 2016

---

Anton Marinó Stefánsson  
200492-3239

---

Hrönn Róbertsdóttir  
211192-2409

---

Sigrún Þorsteinsdóttir  
221187-2619

---

Þórdís Jóna Jónsdóttir  
280292-2309



School of Computer Science  
Reykjavík University  
Menntavegi 1  
101 Reykjavík, Iceland  
Tel. +354 599 6200  
Fax +354 599 6201  
[www.reykjavikuniversity.is](http://www.reykjavikuniversity.is)