Hashtag games and voting system for social media management

Autumn 2014

Arnar Þór Sveinsson
Axel Máni Gíslason
Haraldur Andri Stefánsson
Sólberg Bjarki Valdimarsson

BSc | Computer Science

Instructor: Stefán Freyr Stefánsson

Examiner: Birgir Kristmannsson

T-404-LOKA

School of Computer Science

Abstract

The goal of our project is to simplify the process of creating and managing social media competitions and polls. Our system enables companies and individuals to create different variations of these competitions and polls, through a simple process of filling in a template. To control the system, users use a hashtag (#) for the contest or poll so that our system is able to monitor and manage it. Based on the information filled in by the user, our system takes care of the complicated and time consuming part. That involves tasks like finding out which participants meet the qualifications to win, randomly pick winners and then notify the winners, enabling participants to claim the prizes and show admins statistics relating to their campaigns. Another feature of our system is to allow users to have a single competition or poll running on more than one social media, our system takes care of gathering the data from those social medias and all data from them as one.

Samantekt

Markmið verkefnisins er að einfalda ferlið að búa til samfélagsmiðla keppnir og kannanir. Kerfið okkar gerir fyrirtækjum og einstaklingum kleift að búa til og halda utan um mismunandi útfærslur af þessum keppnum og könnunum í gegnum mjög hentugt og fljótlegt ferli þar sem einungis þarf að fylla út einfalt eyðublað. Til að stjórna kerfinu er notað hashtag (#) og vinnur kerfið svo úr uppýsingum frá notanda og sér um flóknu og tímafreku verkin. Það felur í sér að finna út hvaða þáttakendur uppfylla lágmarks þáttökuskilyrði til að komast í pott, velja vinningshafa af handahófi og hafa samband við þá, gera þeim kleift að nálgast vinninga og birta stjórnendum tölfræði fyrir þeirra keppnir. Kerfið okkar gerir eigundum kleyft að hafa sömu keppni eða könnun á mörgum samfélagsmiðlum, en þó eins og um eina keppni eða leik væri að ræða.

Contents

1 Introduction

For most companies today social media sites, such as Facebook and Instagram, are a huge part of their advertisement and marketing procedures. A popular way for these companies to advertise and draw attention to themselves on social medias is creating contests, polls, quizzes and sweepstakes. These competitions and polls usually involve participants commenting or liking a status to either win a price or express their opinions on the company or its products. But managing these competitions and polls properly is usually difficult and very time-consuming. For example if you created a competition you would need to go through the comments and likes and see whether the participants have fulfilled all the requirements needed to win. You would then have to pick winners randomly, notify them and tell them how to claim their prize. This becomes even more complex and time consuming if you decide to have the competition running on more than one social media site simultaneously.

Our system simplifies this process in a way that the creator of the contest/poll only needs to fill out a simple template and then control that contest/poll using a hashtag (#).

2 Artifacts

The following artifacts are on the attached CD.

- **Final report:** This document
- **Project journal:** An Excel document containing all of our work hours along with distribution over each sprint
- **Work procedure:** A document containing how we organized the project and what tools we use to do it.
- **TMNT-README:** Document which contains information about how to set up the project. This document is the READMEs from our repositories combined into one document.

3 Organization

The project started in August with a meeting with Sesselja where she described the project and what she wanted us to do. After that she gave us keys to the office, the whole product backlog and access to the Tagplay project.

3.1 Methodology

After looking at the product backlog we decided to use the Scrum methodology. Scrum was chosen because of familiarity with Scrum from school. The company uses Kanban but we decided to use Scrum because it fit the project nicely.

We divided Scrum roles between us.

Product owner Sesselja Vilhjálmsdóttir

Companies contact Fannar Snær Harðarson

Scrum master Arnar Þór Sveinsson

The team

Arnar Þór Sveinsson

Axel Máni Gíslason

Haraldur Andri Stefánsson

Sólberg Bjarki Valdimarsson

We broke the project into eight sprints. In the beginning we choose roughly what stories would go into each sprint, but at the beginning of each sprint we organized the next sprint beforehand to take in the most important stories.

We started at sprint 0, which went mainly into getting familiar with existing Tagplay code and set-up. We decided to have 8 sprints, excluding sprint 0. The first 6 sprints were two week sprints but during the last two sprints we switched to one week sprints.

After seeing our timetables we decided to work on Tuesday, Wednesday and Thursdays and on other days if there weren't any school assignment. To keep track of this we used Google Calendar (see Time recording and management on the CD).

Before each sprint we had a small meeting where we talked about what stories should go into the sprint and created a backlog for the sprint. We used story point system for measuring work. During these meetings we also discussed what went well during the sprint we were currently concluding and what we wanted to do better in the next one.

4 Risk Analysis

Many risks can arise while developing big projects and must therefore be noted as they arise. Some risks are discovered during testing and others while designing. The possible risks that could impact the project evaluated by their severity and consequence are listed in the table below. Below O stands for odds, S for severity and RF for risk factor. Values of O and S are given on the interval 1-5.

#	Event	O	S	RF ↓	Status
1	**Wrong evaluation of sprints and user stories** Each user story could be miscalculated and as a result each sprint could be either too big or too small. **Reaction:** In each sprint review we will try to re-estimate stories that are similar to those that we miscalculated and if necessary reevaluate the next sprint according to the new calculations of stories. **Consequences:** Extra time needed for sprint planning. **Team member responsible:** Arnar Þór **Occurred:** During sprint 4. **Resolution:** We miscalculated the sprint in the sprint planning, and realized it the morning after so we decided to reorganize the whole sprint.	5	2	10	Resolved
2	**Too much workload in other classes** Other classes might have large assignments that we need to hand in during a sprint. **Precautionary actions:** Watch Myschool closely so that we see the assignment as soon as the due date is set. **Reaction:** We need to plan/rearrange the sprints that these large assignments collide with and take time away from this project to work on and hand in those assignments. **Consequences:** Loss of work hours for project. **Team member responsible:** The team members who are taking a given course in which the assignment is due. **Occurred:** N/A **Resolution:** N/A	3	3	9	Avoided

#	Event	O	S	RF ↓	Status
3	**Poorly implemented features on our behalf** A certain feature that we implement, and needs to be used through out the system, is poorly implemented. **Precautionary actions:** Take time to think what the best implementation would be. **Reaction:** We refactor the feature that is poorly implemented. **Consequences:** Extra time needed to implement feature. Opening an already closed task. **Team member responsible:** Arnar Þór **Occurred:** During sprint 3. **Resolution:** We created new tasks which we put into sprint 4 about refactoring certain features.	3	3	9	Avoided
4	**Poorly implemented features on behalf of the company** A feature that exists in the system, that our system relies on, is poorly implemented. **Precautionary actions:** None **Reaction:** Case 1: We know the feature and how it works - One of the team members refactors the feature for the company. Case 2: We do not know the feature or how it works - We contact the company and ask them to refactor the feature. When the company has refactored the feature they would notify us and we would keep working on the task that was depending on this feature. **Consequences:** Extra time needed to implement feature. Possibly needing to drop a task for a certain sprint. **Team member responsible:** Arnar Þór **Occurred:** During sprint 3 **Resolution:** The Facebook callback implementation that tagplay had was poor so we reimplemented it for them which therefor took us longer than expected.	2	4	8	Resolved

#	Event	O	S	RF ↓	Status
5	**Problem with heroku server** Heroku server works differently than other servers being used by Tagplay and therefor it would need a setup phase and debugging session. **Precautionary actions:** None **Reaction:** Team member responsible for this action would contact the company and get assistance in setting up all the necessary services. **Consequences:** Extra time needed to fix server before being able to work on tasks **Team member responsible:** Arnar Þór **Occurred:** During first sprint **Resolution:** Extra work hours with company CTO in order to get things up and running.	2	4	8	Resolved
6	**Problem with social media APIs** Social media API's do not offer the services that we need in certain tasks. This could for example be not being able to send private messages through apps, not being able to read certain data. **Precautionary actions:** None **Reaction:** This would require a meeting with the company in order to redesign the tasks that this would affect. **Consequences:** Extra time needed to design feature. Could lead to dropping tasks if there is no other way of implementing it. **Team member responsible:** Task assignee. **Occurred:** During sprint 2 and sprint 8. **Resolution:** Sprint 2: Implementation design was changed to fit the Social Media API, the new design ended up being even better than the first design. Sprint 8: Instagram media API was constricting when it came to commenting winners. We solved it by adding a new view, where people can see if they won.	2	4	8	Resolved

#	Event	O	S	RF ↓	Status
7	**Team members getting sick** Team members are unable to work because of health issues **Precautionary actions:** None **Reaction:** Everyone that are not sick try to pick up the slack for the person that is missing. If everything goes well this will not affect the sprint. If something was not finished we would need to mention it in sprint review and alter the next sprint accordingly. **Consequences:** More workload on other team members. Sprint tasks would be delayed. **Team member responsible:** Sólberg **Occurred:** N/A **Resolution:** N/A	3	2	6	Avoided
8	**Severe arguments between team members** Team members do not agree on various things, such as design, workload, etc. **Precautionary actions:** Talk things out before they become severe **Reaction:** Get both parties to make compromises and find a common ground that everyone can agree on. **Consequences:** Time spent solving argument **Team member responsible:** Axel **Occurred:** N/A **Resolution:** N/A	1	4	4	Avoided

#	Event	O	S	RF ↓	Status
9	**Communication issues with company** Product owner or company contact go abroad, become sick or lose some of their contact with our project. This could delay acceptance of new designs, prioritising tasks or server issues. **Precautionary actions:** Try to be active in talking to the company contact and be active on gitter and sending emails **Reaction:** The team will have to use other communication, such as skype, hangout, gitter and email to communicate with product owner and/or company contact. **Consequences:** Delaying tasks until company replies. Delaying tasks until company accepts new design of a feature. **Team member responsible:** Arnar Þór **Occurred:** After first sprint **Resolution:** Product owner (Sesselja) went to the US until christmas. Instead of talking to her at the office, asking her about decision, we will use emails, gitter and skype. On demo days, we will use skype to communicate and show her our product.	2	2	4	Resolved
10	**Team members computer crashes** Team member's computer crashes or there are some bugs with the software. **Precautionary actions:** Keep all software updated. Try to push frequently new code to github, so will not lose any code. Keep all reports on dropbox and time logs on Microsoft Office 365 **Reaction:** Detect the problem, if it is a software problem install the required software or fresh copy of the operating system If it is a hardware problem replace the broken one. **Consequences:** Extra workload on other team members during repairs **Team member responsible:** Sólberg **Occurred:** N/A **Resolution:** N/A	1	3	3	Avoided

#	Event	O	S	RF ↓	Status
11	**Server and/or network problem at final presentation** Production server is down/slow or there is problem to connect to it on the final presentation. **Precautionary actions:** Take screenshots of all views and pages beforehand to show what the system looks like and should work. **Reaction:** Open the slideshow with the screenshots and smile. **Consequences:** The final presentation will be less impressive as we can not show the flow of the system. **Team member responsible:** Axel **Occurred:** N/A **Resolution:** N/A	1	2	2	Avoided

5 Architecture

This design report gives you an overview of the Tagplay campaign system. The campaign system is used to help companies create, manage and automate campaigns on their Social Media accounts. Companies will be able to create a campaign, specify the rules to participate (ex. What is the capital of England. Comment with the correct answer and like the post and you can win a trip to the capital.), specify prizes and when the deadline for the campaign should be. Our system would then automate the rest. We would gather all answers, filter out which participants are potential winners and even send a message to those that have won when the deadline is over.

5.1 Component Details

5.1.1 Midgard

Overview: Is the frontend, a part of it had already been written by Tagplay but everything regarding campaigns and sharing to social media is written by us. There are two possible users that use Midgard, the campaign owner and a campaign participant. The campaign owner logs into his account and creates a campaign. The campaign participant authenticates himself through a link that is sent to the winners of the campaign and can then see details about his prize and claim it.

Our additions: We will implement into this component an overview of all campaigns a certain user has, statistics for each campaign and detailed information about each campaign (ex. How many participated, who won, what was the prize for this campaign etc.). We will also implement a form which users can fill out to create a new campaign.

Talks to	Through
Tagplay server	HTTP

Table 7: Midgard communication

5.1.2 Tagplay Server

Overview: Is the backend, like Midgard a part of it had already be written by Tagplay but everything regarding campaigns and sharing to social media is written by us. It receives http requests from Midgard and either stores in PostgreSQL database or forwards it to the appropriate service through the message queue.

Our additions: We will add all functionality regarding campaigns. This includes the data models, REST API, reading comments from social media and storing relevant data for participants and picking winners. We will also add to the formula functionality already in the Tagplay system regarding formulas that share from one social media to another.

Talks to	Through
Services	MQTT/Json
PostgreSQL	Django ORM
Connection stores	HTTP

Table 8: Tagplay server communication

REST URI: Bellow is list of all REST URIs on Tagplay server that we will make.

Type	**URI**	**Description**
GET	/api/v1/campaign/	Retrieve a list of campaigns
POST	/api/v1/campaign/	Create a new campaign
GET	/api/v1/campaign/id/	Retrieve a single campaign by ID
PUT	/api/v1/campaign/id/	Update an existing campaign
DELETE	/api/v1/campaign/id/	Delete an existing campaign
GET	/api/v1/connection/	Retrieve a list of connections
POST	/api/v1/connection/	Create a new connection
GET	/api/v1/connection/id/	Retrieve a single connection by ID
PUT	/api/v1/connection/id/	Update an existing connection
DELETE	/api/v1/connection/id/	Delete an existing connection
GET	/api/v1/participant/	Retrieve a list of participants
POST	/api/v1/participant/	Create a new participant
GET	/api/v1/participant/id/	Retrieve a single participant by ID
PUT	/api/v1/participant/id/	Update an existing participant
DELETE	/api/v1/participant/id/	Delete an existing participant
GET	/api/v1/prize/	Retrieve a list of prizes
POST	/api/v1/prize/	Create a new prize
GET	/api/v1/prize/id/	Retrieve a single prize by ID
PUT	/api/v1/prize/id/	Update an existing prize
DELETE	/api/v1/prize/id/	Delete an existing prize
GET	/api/v1/submission/	Retrieve a list of submissions

POST	/api/v1/submission/	Create a new submission
GET	/api/v1/submission/id/	Retrieve a single submission by ID
PUT	/api/v1/submission/id/	Update an existing submission
DELETE	/api/v1/submission/id/	Delete an existing submission
GET	/api/v1/winner/	Retrieve a list of winners
POST	/api/v1/winner/	Create a new winner
GET	/api/v1/winner/id/	Retrieve a single winner by ID
PUT	/api/v1/winner/id/	Update an existing winner
DELETE	/api/v1/winner/id/	Delete an existing winner
POST	/shareto/service	Create a new share to service
GET	/campaign/facebook/authenticate	Authenticate facebook user
GET	/campaign/facebook/callback	Get access token from facebook
GET	/campaign/instagram/authenticate	Authenticate instagram user
GET	/campaign/instagram/callback	Get access token from instagram
GET	/campaign/id/winners	List all winners for a certain campaign

Table 9: REST URIs on Tagplay server

5.1.3 Connection Store

Overview: Had already been written by Tagplay, it keeps track of all the connections to social media accounts and stores them in LevelDB.

Our addations: None

5.2 Services

5.2.1 Share Service

Overview: Listens to the message queue and shares to social medias accordingly. This service gets information about all formulas from the Tagplay server and stores them in Redis. When the Tagplay server recieves new media it will publish to the message queue information regarding that. This service will read that information and check if it should be shared any further which would then be done through the relevant Social Media API.

Our additions: This component is written entirely by us.

Talks to	**Through**
Social Media APIs	HTTP
Tagplay server	MQTT/Json
Redis	Redis client

Table 10: Share service communication

5.2.2 Campaign Service

Overview: Listens to the message queue for messages regarding callbacks from social medias. This checks if the post should be part of a campaign and creates submissions.

Our additions: This component is written entirely by us.

Talks to	Through
Social Media services	MQTT/Json
Redis	Redis client
Tagplay server	HTTP

Table 11: Campaign service communication

5.2.3 Social Media Services

Overview: We've got one service for each social media (facebook, instagram, twitter and tumblr), they all listen to the message queue and send http request to their social media api depending on what is published to the message queue. These services also take care of normalizing all data they get from the social media APIs and sends it to the tagplay server to store it.

Overview: The Facebook, Tumblr and Instagram services will be written entirely by us but the Twitter service exited already and does not need modification.

Talks to	Through
Social Media APIs	HTTP
Redis	Redis client
Connection store	MQTT/Json
Tagplay server	MQTT/Json

Table 12: Social media services communication

5.3 Overview

Figure 1 shows a overview of all components in the system and how they communicate between each other. For convenience we have shortened social media to SM and message queue to MQ in our sequence diagrams.

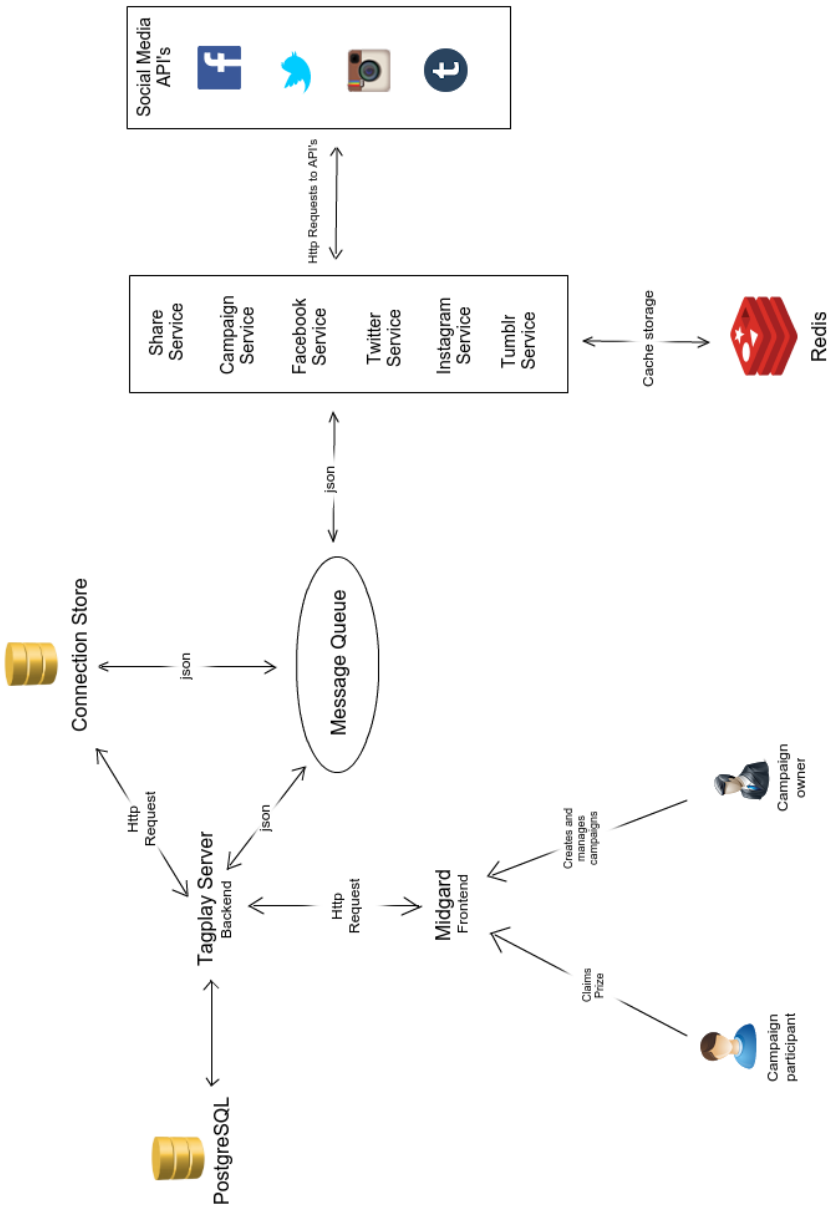

Figure 1: System overview

5.4 Sequence diagrams

Figure 2 shows a sequence diagram of a Tagplay user creating a new Campaign. The diagram shows how the message cascades through all relevant components to make sure each relevant component knows about the new campaign.

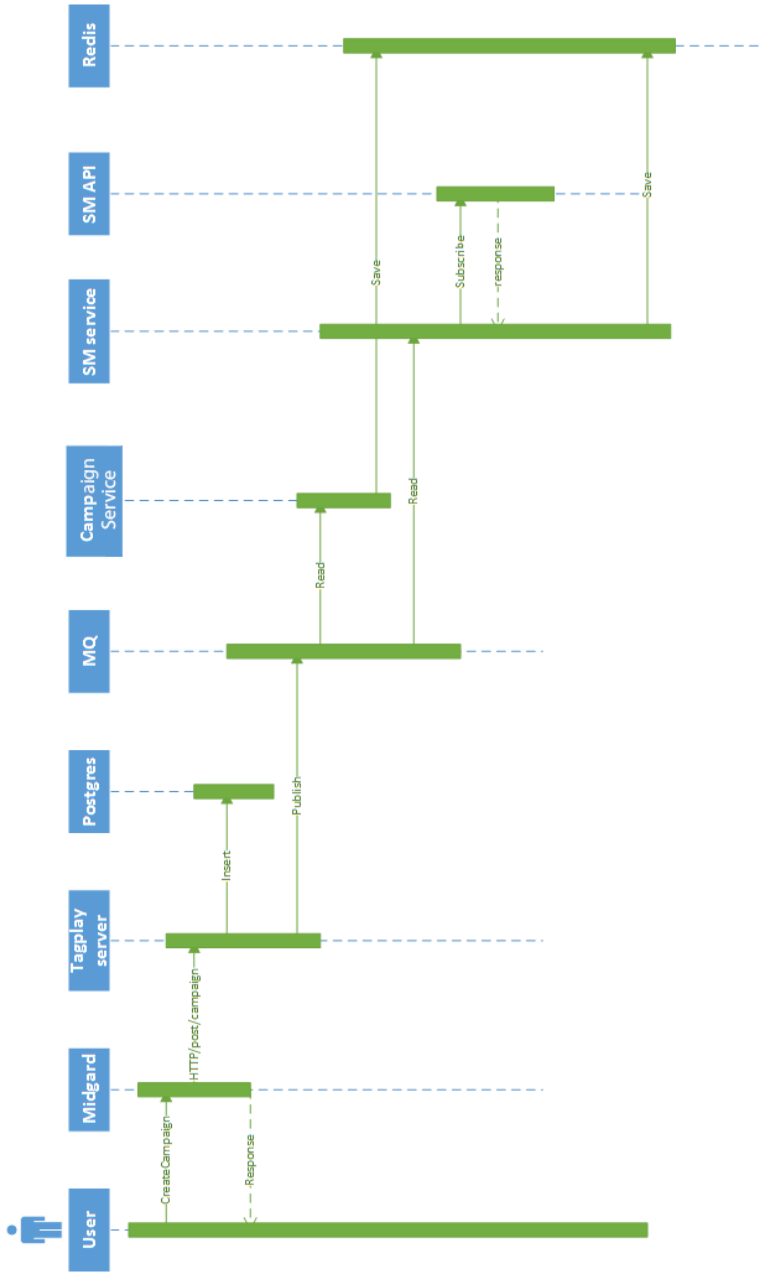

Figure 2: Sequence diagram: User creates a new campagin in the Tagplay system

Figure 3 shows a sequence diagram of a Tagplay user that already has a Campaign in Tagplay and posts a new status/tweet/image to a Social Media. The diagram shows how the realtime callback reaches the relevant Social Media service which parses the newly posted media and if it's part of any Campaign in our system it will send out a message which cascades through the components until it reaches the database in the Tagplay server.

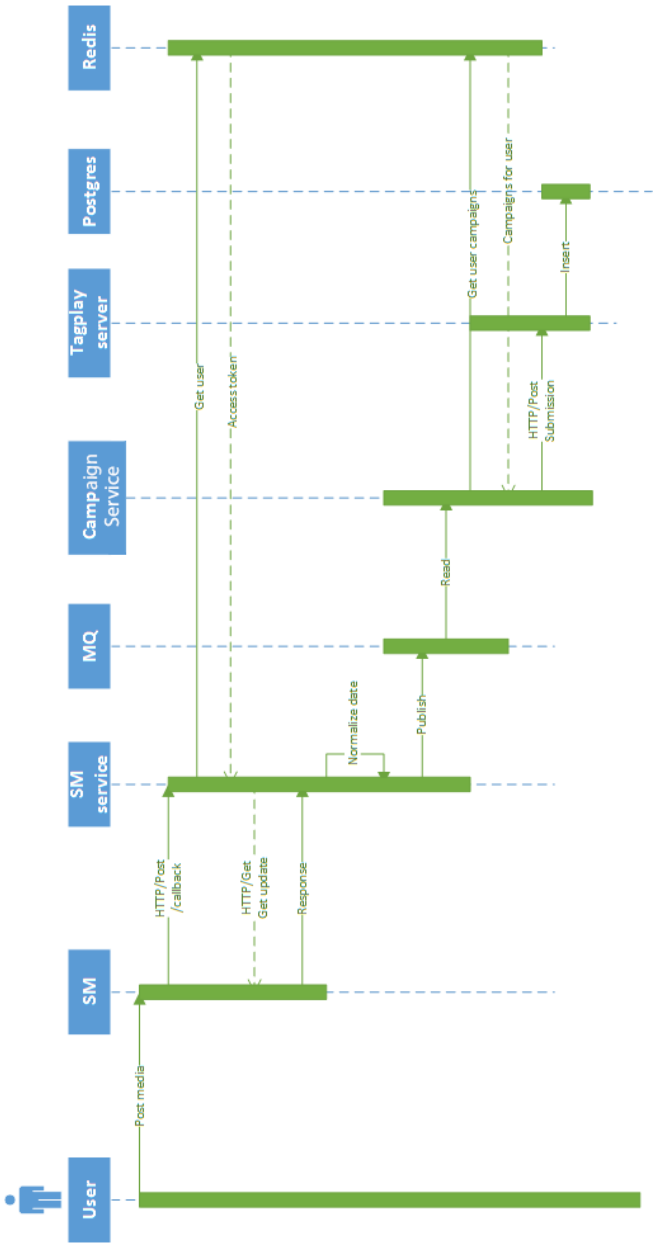

Figure 3: Sequence diagram: User posts media (status, tweet, image etc.) on social media

6 Progress

The initial plan was to work around 150 hours for eight weeks or 1200 hours in total. During the first sprint, we realised that we needed to raise the working time for each sprint and reconsider how much each story point weighted. It was clear early on that we would not finish all the stories we set up with, so we decided to cut the less important stories in cooperation with the product owner.

6.1 Sprints

6.1.1 Sprint 0 - Pizza time

> "Mm-hm. Pizza dude's got thirty seconds."
>
> — Michaelangelo

Our goals for this sprint was to start planing the project and set up all required software. Additionally we started taking a look at existing code from the company that we where going to be working with.

This sprint was quite good. We had a meeting with the company, where they explained the project in details for us. Then they gave us keys to the office so we could set up our working station and have full access to the office.

When all stories had been converted to Github issues we organized Sprint 1 and made a rough schedule of all other sprints, which will be altered in each Sprint review.

6.1.2 Sprint 1 - Facing the Shredder

> "COWABUNGA!"
>
> — Leonardo, Michaelangelo, Raphael, Donatello

Our goals for this sprint was to start programming and get familiar with the Tagplay system. On the whole we were pleased with this sprint. We managed to finish our most

important user stories and on demo day the product owner was pleased with the product so far. We had some issues with setting up and configuring servers and therefor too much time went into debugging in our opinion. We also realized that the learning curve for ReactJS was steeper then we anticipated and we suspect that we will notice this in future sprints. We also saw that we underestimated some stories and overestimated others. Also we had to take some time for reports. We added one story when Tagplay changes it system to use message queue(story #111).

We learned a lot in this short period of time and we should now be more adept at estimating and planning sprints from now on. There were however some stories that we didn't have time to finish and a few that we didn't start working on. These stories are important for the beginning of the project and will therefore be moved into the next sprint.

#	**User Story**	**Points**	**Status**
#7	Formulas can have share action as well as send to action	6	Closed
#8	Search status for share tags	1	Closed
#9	Search status for no-share tags	1	Closed
#11	Formula deadline	2	Closed
#12	Coupon code	2	Closed
#13	Send coupon code for facebook	2	Closed
#14	Authentication logic for facebook	4	Closed
#16	Sweeptakers winner limit	2	Closed
#17	Sweepstakes (enter/gimme) template	6	Open
#19	Collect form data	2	Open
#20	Giveaways form	2	Open
#21	Inactivate formula with comment for facebook	2	Closed

#22	Coupon/Discount/Gift card template (X many)	6	Open
#23	Winner pick with random draw	2	Open
#61	Search post for no-share tags	1	Closed
#111	Create share MQ service	6	Closed

Table 13: Backlog for sprint 1

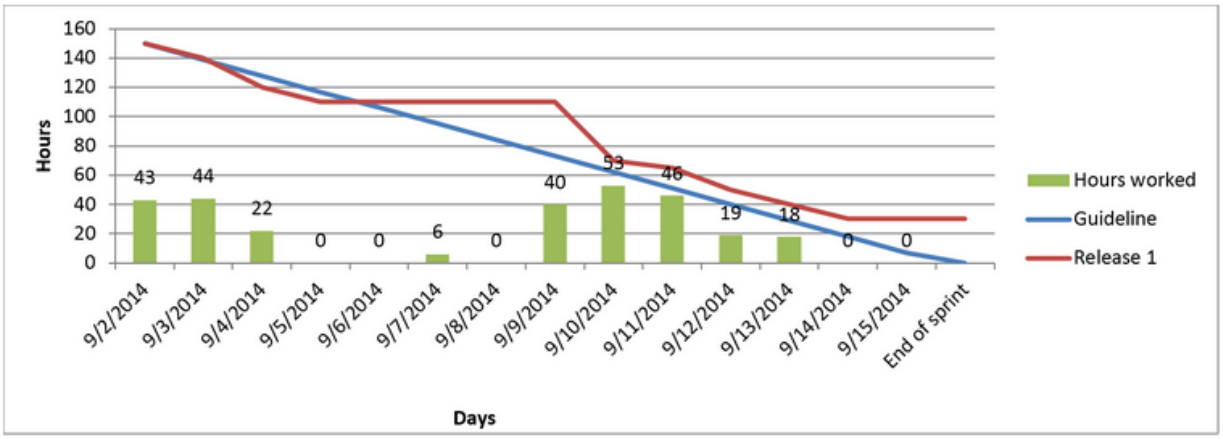

Figure 4: Burndown chart for sprint 1

6.1.3 Sprint 2 - Reborn

> "There will be no mistakes this time... I go myself. And the rat - kill it."
>
> — Shredder

Our goals for this sprint was to get back on track and reduce the number of mistakes. This sprint was uneventful compared to the first one due to almost everything going according to plan.

What we learned in sprint 1 really showed during the organization of this sprint. We realized that we needed to count the number of story points we put into each sprint instead of estimating ourself. We thought that since we were more familiar with the code

and Tagplay system we could take in same number of stories point as in the previous sprint and make the sprint larger according to the story points that went in.

We cancelled four stories all of which had to due with that their social media API didn't offer what we needed. We took one extra story in regarding Facebook service (story #120). We finished every story except one in this sprint. Although we worked less then we estimated we finished almost everything due to the fact that we had started some stories in the previous sprint.

#	**User Story**	**Points**	**Status**
#17	Sweepstakes (enter/gimme) template	6	Closed
#22	Coupon/Discount/Gift card template (X many)	6	Closed
#24	Winner pick from correct answers for facebook	6	Closed
#25	Giveaways and sneakpeeks	6	Closed
#27	Winner pick limit per user	4	Closed
#33	Submit prize code	2	Closed
#57	Authentication logic for Instagram	6	Closed
#74	Automated messaging direct message for facebook	1	Cancelled
#113	Reimplement share-to parser	4	Closed
#114	Share-to for Pinterest feed and boards	4	Cancelled
#115	Share-to for Tumblr Public and Private Blogs	4	Closed
#116	Share-to for Linkedin Profiles, Groups and Company Pages and Business page	4	Cancelled
#117	Share-to for Google+	4	Cancelled
#119	Winner template	4	Open
#120	Facebook service for Realtime callbacks	4	(Added) Open

Table 14: Backlog for sprint 2

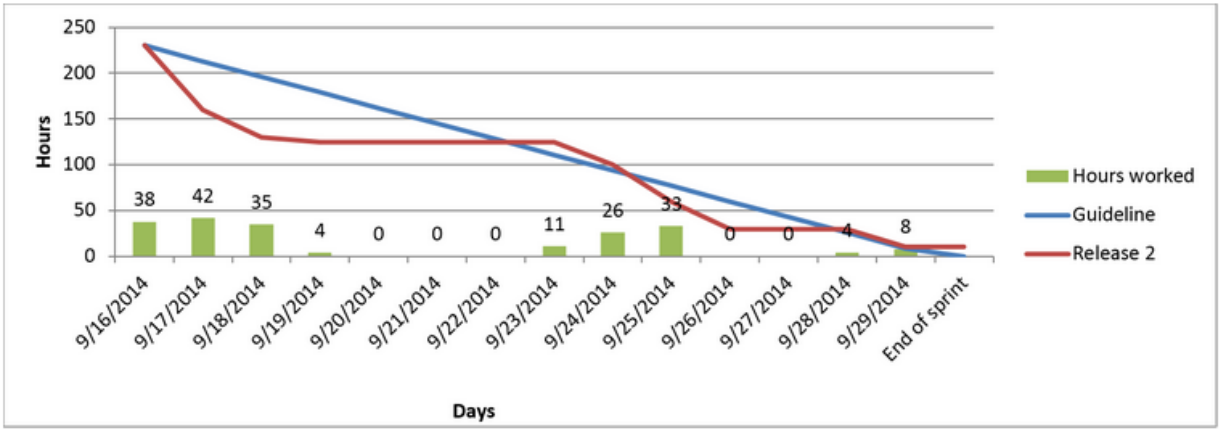

Figure 5: Burndown chart for sprint 2

6.1.4 Sprint 3 - We will find you!

"Yeeeaaah, we'll fiiiind yoooou! O'Neil.!... I'm sorry, that came across super-creepy. We will find you, though!"

— Michelangelo

We decided to make the sprint shorter about third shorter or only 100 hours since we saw on our timeline that a lot of assignments were due this week. Everything went according to plan. We saw that changing each sprint size was no problem if we organize well but we also realized that we took too many stories in. We realized that the size we took into this sprint should be appropriate for a normal sprint.

We added one story to make it easier for us to test and for others to see the campaigns they have (story #121).

#	User Story	Points	Status
#20	Giveaways form	6	Closed
#23	Winner pick with random draw	2	Closed

#28	Participants rules for facebook	4	Closed
#29	Simple comment rules for participants for facebook	6	Open
#34	Submit reply to winners for facebook	4	Closed
#35	Automated comment on deadline for facebook	2	Closed
#119	Winner template	4	Closed
#120	Facebook service for Realtime callbacks	6	Closed
#121	Campaign view	6	(Added) Open

Table 15: Backlog for sprint 3

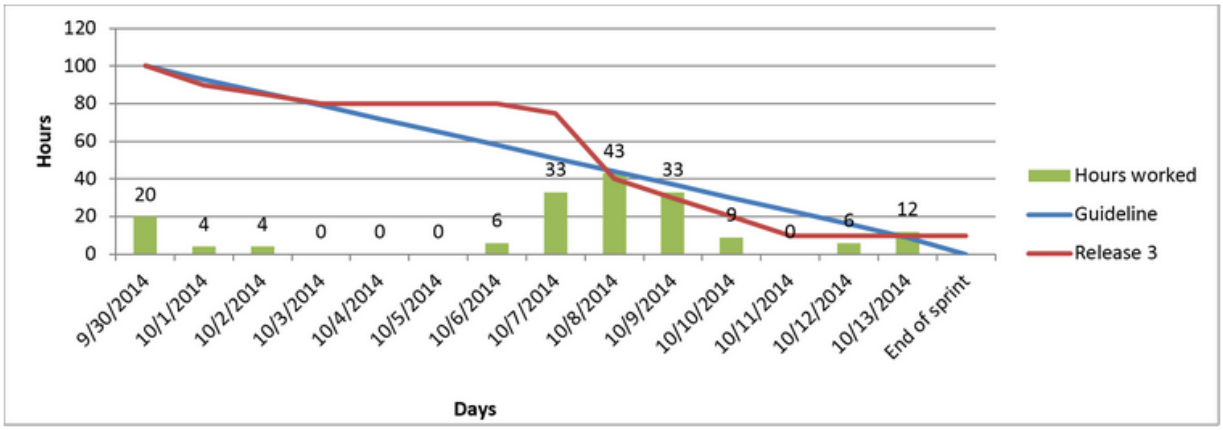

Figure 6: Burndown chart for sprint 3

6.1.5 Sprint 4 - The master

"Remember my son, everything you know I have shown you. But I have not shown you everything I know."

— Master Splinter

We miscalculated the sprint in the beginning but realized it the morning after so we decided to re-organize the whole sprint and take some time to do reports and plan for

meeting. After that, the sprint went fine, we left one story open but we came close to finishing it. We are overall pleased with how everything went.

#	User Story	Points	Status
#29	Simple comment rules for participants for facebook	6	Closed
#76	Automated messaging status update on facebook page	1	Closed
#121	Campaign view	6	Closed
#123	Gather basic information from campaign	4	Closed
#124	Represent campaign information using something like D3	6	Open
#126	Refactor share to functionality	6	Closed

Table 16: Backlog for sprint 4

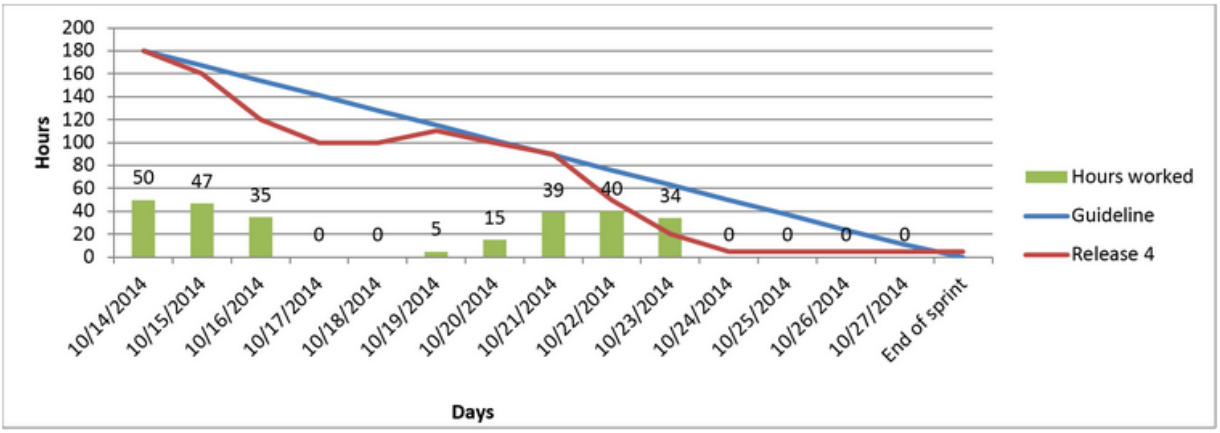

Figure 7: Burndown chart for sprint 4

6.1.6 Sprint 5 - The empire

"Your empire flourishes, Master Shredder"

— Tatsu

Everything went as planned. We added one story since our product owner wanted to get a working product to test (story #127). We finished 11 of the 12 stories in the sprint.

#	**User Story**	**Points**	**Status**
#40	Inactivate formula with comment for Instagram	2	Closed
#41	Winner pick from correct answers for Instagram	6	Closed
#42	Participants rules for Instagram	2	Closed
#43	Simple comment rules for participants for Instagram	6	Closed
#49	Submit message to winners for instagram	2	Closed
#51	Automated comment on deadline for intagram	2	Closed
#91	Email notifications to admin	1	Closed
#104	Voting - limit the pool to one entry per user - Instagram	1	Closed
#122	Create Voting template	6	Closed
#124	Represent campaign information using something like D3	6	Closed
#127	Finalize Campaigns for Facebook	6	(Added) Open
#128	Test sweepstakes for facebook	2	Closed

Table 17: Backlog for sprint 5

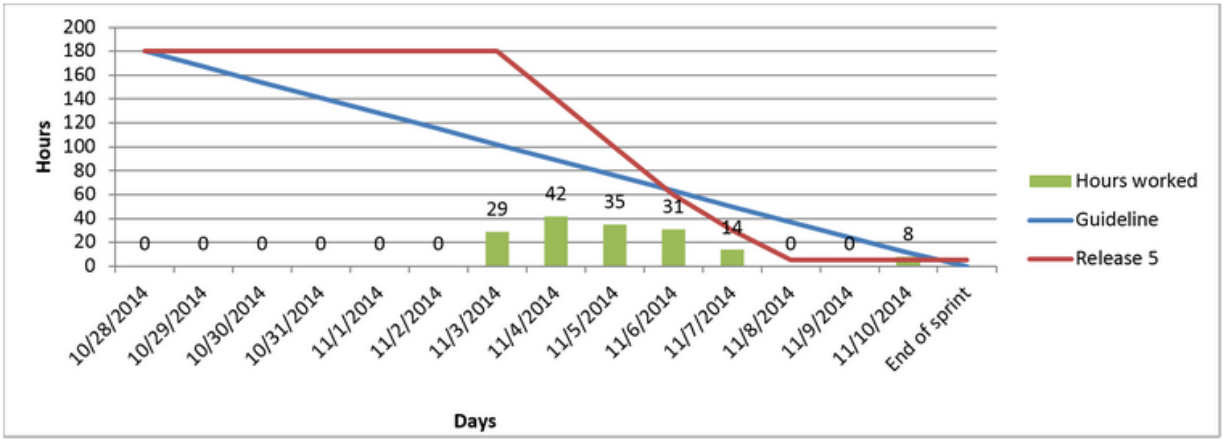

Figure 8: Burndown chart for sprint 5

Sprint 6 - No defeat, no surrender

> "God, I LOVE *BEING A TURTLE*!"
>
> — Michaelangelo

This sprint was shorter then the previous ones excluding sprint 3 because of our final exams. Only two members worked the first week and the other two worked the latter week. Even though the sprint was shorter everything went well and we finished everything we started.

#	User Story	Points	Status
#64	How many people participated in sweeptakers for facebook	2	Closed
#66	As a user I should be able to pick age restirction to participate or get rewards	2	Cancelled
#67	How many liked sweeptakers for facebook	1	Closed
#72	Check if we can use twitter cards	1	Cancelled
#82	Direct message for instagram	1	Cancelled
#93	How many people participated in sweeptakers for instagram	2	Closed

#95	How many liked sweeptakers for instagram	1	Closed
#97	Participation rate for sweeptakers for instagram	6	Closed
#129	Fix backend for new format of data	4	Closed
#137	CSS in campaign view	2	Closed
#138	Fix worker problem in backend	6	Closed

Table 18: Backlog for sprint 6

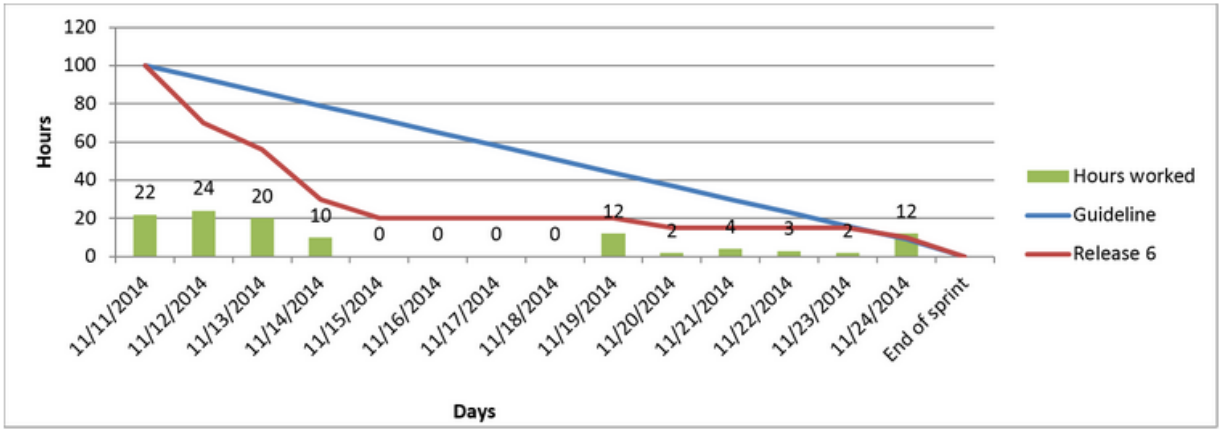

Figure 9: Burndown chart for sprint 6

6.1.7 Sprint 7 - Back at the streets

> " It's time for us to go back!"
>
> — Leonardo

In this sprint we changed our plan from previous sprints. We work everyday but the sprint is only a week long. Most of the time in the sprint went into fine tuning and debugging features that we had already programmed as well as redesigning our front end. We also merged Tagplay master branch into our master at their request. This caused some things we had done changed to not work which added some extra debug on stories.

#	User Story	Points	Status
#19	Collect form data	2	Cancelled
#59	Authentication site	6	Closed
#127	Finalize Campaigns for Facebook	6	Closed
#131	Word cloud statistics	4	Closed
#132	Refactor doughnut chart for voting	2	Closed
#133	Fix backend to take in multiple answers	2	Closed
#134	Debug instagram submission	2	Closed
#135	Test sweepstakes from two socials media	2	Open
#136	Add open/closed voting questions to backend	6	Closed
#139	Refactor campaign template	6	(Added) Closed
#153	Merge Tagplay back and frontend into our project	4	Closed

Table 19: Backlog for sprint 7

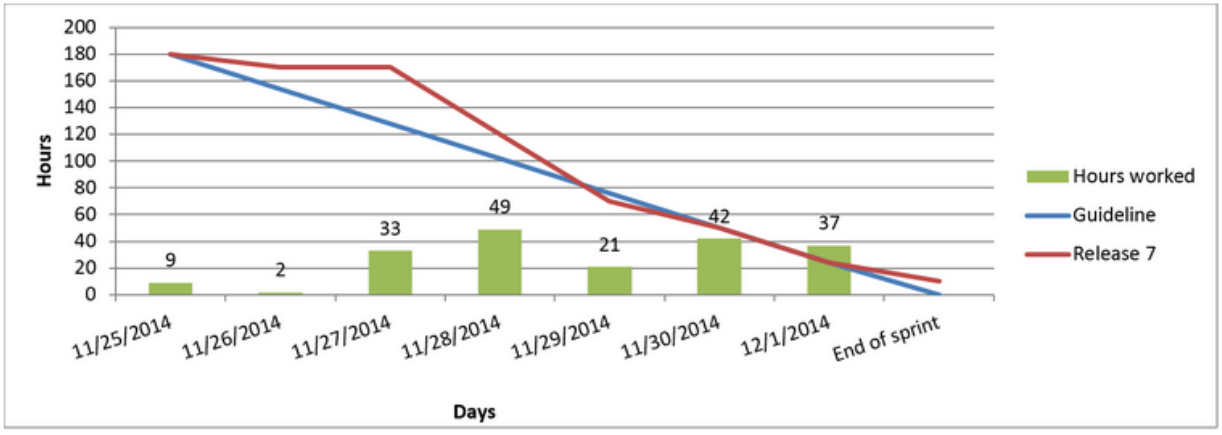

Figure 10: Burndown chart for sprint 7

6.1.8 Sprint 8 - Turtles Forever

> "You see me as I am, and it will be the last thing you will ever see."
>
> — Shredder

This sprint as the previous one was all about fine tuning and fixing bugs we knew of. We hit a small wall when a story we thought was finished resurfaced so we had to add a story about fixing the message queue bug where it subscribed everything twice.

#	**User Story**	**Points**	**Status**
#135	Test sweepstakes from two socials media	2	Closed
#141	Refactor stats page	6	Closed
#142	Delete winning prize for everyone else	2	Closed
#143	Validate campaign template	4	Closed
#144	Fix replies on posts (worker/MQ problem)	6	Closed
#145	CSS on stats page	4	Closed
#146	Generate fake data	2	Closed
#147	CSS datetimepicker	2	Closed
#148	Datetimepicker on campaign template	4	Closed
#149	Validate state page	2	Closed
#150	Fix share-to	4	Closed
#151	Debug frontend after merge	4	Closed
#152	Debug backend after merge	4	Closed

Table 20: Backlog for sprint 8

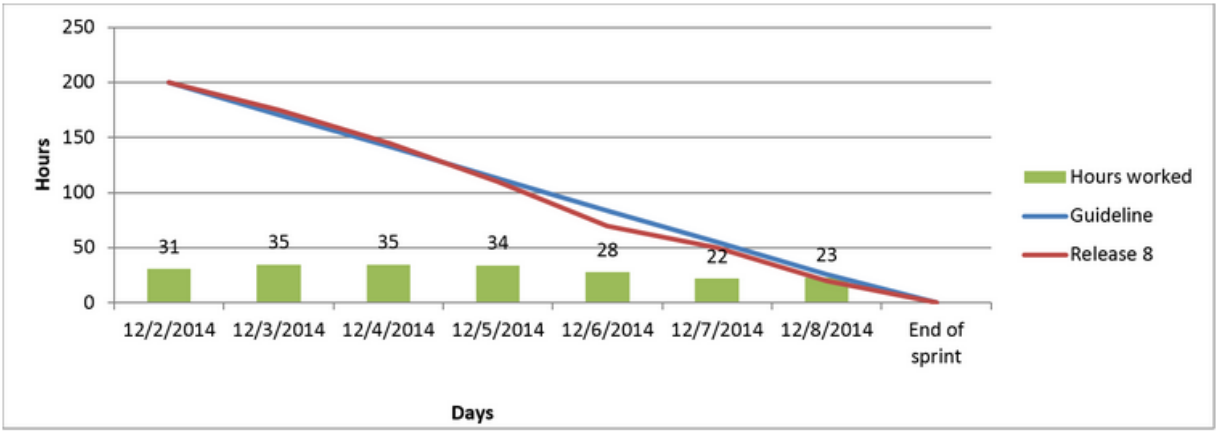

Figure 11: Burndown chart for sprint 8

6.2 Sprint summary

Figure 12 shows the overview of the overall process. As can be seen the curve is almost linear and according to the guideline. We didn't implement all feature we set up to begin with simply because of time was limited and thus the less important features in product owners opinion were left out.

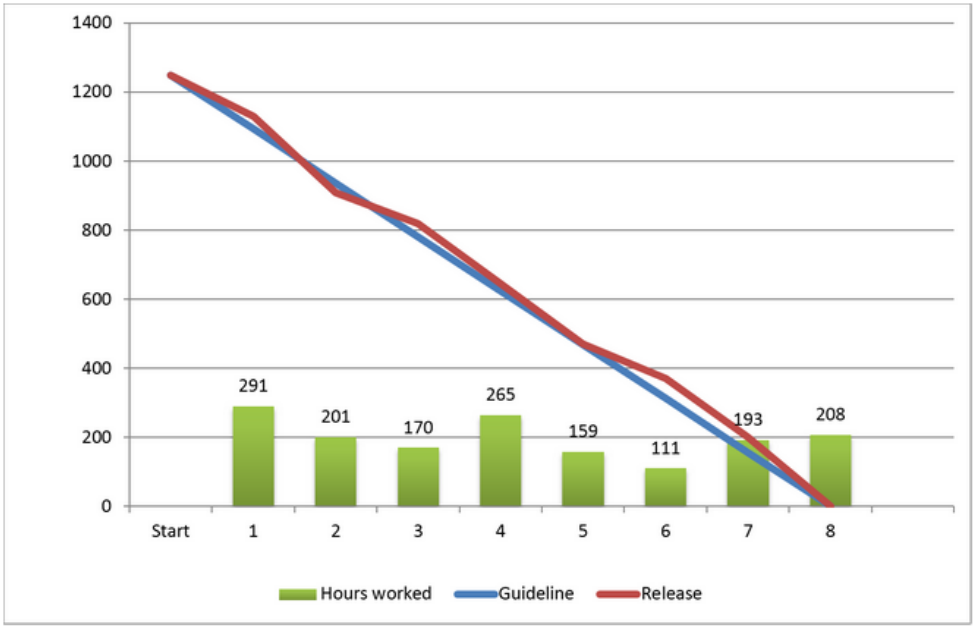

Figure 12: Project burndown

Below in table 21 the time spent on the project can be viewed. A more detailed overview of hours worked for each sprint can been seen in *Time Journal* document on an attached CD. There the hours worked by each team member on each sprint can be viewed as well as the summarized time.

Name	**Estemated work time**	**Actual work time**	**Total**
Arnar	330	477	145%
Axel	330	369	112%
Haraldur	330	351	106%
Sólberg	330	399	121%
Total	**1320**	**1596**	**121%**

Table 21: Total time summary

7 Summary

Overall we are very pleased with the outcome of the project even though the scope of the project was extensive. We are happy with the teamwork and enthusiasm from the team as a whole. Each team member was really interested in the project which made working together and planning easy.

There are some things that could have been better. We underestimated some stories and overestimated others, but in the end that did not effect the final result. Some time and knowledge in ReactJS beforehand and getting to know the system would have been convenient since the learning curve was quite steep, and quite alot bigger then we anticipated.

We learned a lot of things during the semester and we are certain it will help us on future projects. It was clear, as we worked on this project, that programming skills were not the only essential thing to finish it. It also took good teamwork, communication and planning. Understanding of Scrum and how it can help a development of a project like this also became a lot clearer.

We would like to thank Sesselja for giving us the chance to work with something so new and exciting, Fannar Snær for being our go-to guy and Stefán for always being happy to give us his opinion.

8 Future

Since the beginning of the project we have been working very closely with Tagplay. We have been working in their repositories on our own branches.
To keep everything updated we have merged their master into ours on a monthly basis to make sure that merging our project into theirs will be as simple as possible.

To help them sync their project with ours we have put a file TMNT-README.md into the 2 main components, Midgard (frontend) and Tagplay-server (backend) with instructions on how to sync their master with ours. (These documents have been put together into one file attached on the CD as TMNT-README)

When these 2 components have been merged all other service will be as simple plug and play since they are already configured to work with the components in our state.

Next steps for the company will be to fix a problem we had during development where we had to change the server to only use one worker thread. There are two possible ways to fix it, either create a worker queue or change the message queue from Mosca to RabbitMQ.