Software Engineering Product Architecture Final

Juicy Story Project Group
Hichem Bouakaz (TA)

Tolga Parlan
S3051072

George Argyrousis
S3175243

Alina Matei
S3080722

Casper Schmit
S3194299

George Rakshiev
S3015564

Wim de Jager
S3215210

Razvan A Poinaru
S2914751

June 11, 2018

Contents

1 Introduction

Nowadays, more and more ads are appearing to consumers in a variety of different ways and companies are increasingly feeling the pressure to distinguish themselves. This is where Juicy Story comes into play, Juicy Story is a web based application that facilitates companies to reach their customers in a more personalized and targeted manner using the Instagram stories feature. Juicy Story allows companies to easily select, schedule and post stories received from third parties / customers via DM's (direct messages) on Instagram. Also, Juicy Story gives an comprehensive overview of the effectiveness of the posted stories with statistics obtained from the Instagram API.

The web application can be viewed as two semi-separate parts, the front end and the back end. Which have been merged together, but can be tested independently. This document describes in detail how the system works and reasons for why certain decisions were made.

2 Back-End

2.1 Overview

The entire back-end has been re-factored from the ground up to facilitate the fully authenticated Django REST API. We have retained relevant functions from the previous implementation but the majority of the code is geared towards making all relevant API calls.

2.2 Technology Stack

- We are using **Django**, a web framework based on Python which allows for seamless front and the back end interaction. The framework supports a modernized version of the MVC pattern called **MVT**.
  MVT stands for (M)odel, (V)iewer, (T)emplate and it's slightly different from MVC. Django alleviates the need for a Controller which is being taken care of internally. The template acts as the front-end while the Model and View both act as the back-end.
- Django REST was the obvious framework of choice. It works seamlessly with Django providing low lever access to authentication, serialization and APIViews allowing us to not worry with the intricacies of a REST framework allowing us to focus our efforts to the data and security of the information that can be requested.

2.3 Django Apps

Re-usability is crucial for any development project and Django provides modules that can be created called 'apps'. The apps can then be reused for multiple projects with slight modifications. So far we have created numerous apps that handle certain aspects of the web-application :

- authentication : This is the django app that is concerned with authenticating with the Instagram API, which is done in a relatively complicated manner. First an authentication request is sent to the API, which includes information about our application, and a redirecting url. Instagram then handles the user log-in, and redirects the user to the passed url, with the GET request including a code. This code can then be used to get an an access token, which will be used in all future API requests concerning this user. While complicated, this whole process happens in the background for the user (except logging into Instagram if their cookies don't have any previous session information), and once the access token is acquired it can be used indefinitely thus we don't need to repeat the process. We do this by saving the access token in our database for each instagram user when applicable.
- database : This is where the model for the whole project is. Sqllite is used for the database. From this app other apps can access the objects stored in the database. At the moment we store images to be uploaded and the Instagram user information, including attributes such as the access token and user name. The database has been extended to hold image urls that make up an instagram story. The action of adding an Instagram account and showing the list of all Instagram accounts managed by the user are handled by the api points existing in this app.
- djangocron : This is an open source django app that allows us to use unix's own cron with django in order to periodically check if there are any uploads scheduled. Basically a cron job notifies the djangocron every minute, which then checks for an upload scheduled to now, or to the past in the database. A further improvement could be to migrate to anacron which is similar to cron but with better support for missed checks during the times that the system wasn't running.
- core : This app contains all base url components that become more complex in the urls.py file of every module that it is tied to. The only url that is directly wired through the core module is the 'home/' url which connects the base html component that can be imported by all other modules of the project. Finally, all the API routers are also included and can be collectively displayed in the 'api/' url. The full api list can be seen in the routers.py file in the core directory.

- incoming : The application responsible for retrieving and displaying Instagram data. The module can directly retrieve content from a predefined Instagram account such as the Instagram user's image, the account name, likes, comments. Currently this module includes search functionality where images can be retrieved and displayed that have a specific hashtag or location tied to our account. Later we would refactor the search functionality and we are going to show a list of all user accounts that have sent at least one picture to the connected account via direct message. Thus the retrieval of images would be contained to the received images only via direct message containing information about the sender and the date.
  A sub-part of Incoming works in conjunction with the front-end allowing the user to select multiple images that can be then saved to the database for later use. The model includes the image url and the Instagram user that is currently selecting the images.

- entry : The application responsible for User management. The actions of logging in and signing up are integrated into the user module. The core design choices that we made here were solely based on the attributes that User would have in a database table that would act as the Model of the application. We decided that the model should only contain information about the name of the user and the password required to access the account. Later iterations would allow user creation under predefined prerequisites, as well as more data stored about the users in accordance with the possible customer demands.

- upload : The app responsible for overseeing the uploading of stories. This app also handles the page that lists the images that are marked for uploading for the current user. This is done by a simple database query. The uploading part is much more complicated since there is no official way to do this through the Instagram API. Much of our code relies on open source libraries doing similar things, and also mimicking the requests that are being sent to the Instagram website when uploading an image by hand. In the future we are planning to migrate the code to use one of such open source library, however that library is still implementing story uploads, thus this can take some time.

- statistics: The statistics app handles all the Instagram user statistics such as reach,followers,views, and etc. In later versions user story statistics will also be added to this app. Statistics are retrieved via the Facebook graph api. Since certain statistics only can be reached for a certain time frame(ie day,month, or lifetime), the retrieval of the statistics is split into 3 functions based on the time frames that are accepted. Using a simple get request the stats are retrieved and rendered to the template file. For the request to go through the user's instagram business id is needed as well as a Facebook access token which differs from the one used for Instagram's api. The functions also use time propagation meaning a from and until date must be given using unix timestamps. In future versions a redirect will be added to the app so the fb access token and Instagram business id can be retrieved and stored in our database.

3 Front-End

For the front end development we decided on using the following technology stack:

1. **HTML & CSS**: HTML is the markup language used to describe the structure of the web page and CSS describes the layout/presentation of the web page.
2. **Bootstrap**: HTML, CSS & JavaScript front-end framework to design responsive websites and web applications.
3. **Vue.js**: A JavaScript framework to build user interfaces.
4. **Javascript (jQuery)**: A lightweight programming language, commonly known as the scripting language of web pages.

3.1 Architectural and design decisions

For the second sprint, we wrote a demo which includes the Login page, a Sign up page, a Password recovery page for resetting the password and a first version of the home (dashboard) page.

The design of the pages follows the guidelines given by the client, however it could further be changed/improved according to the client's requests.

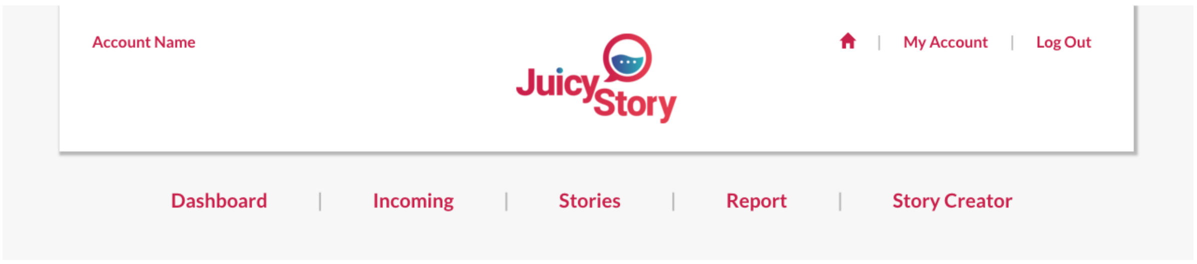
For the third sprint, we further added pages for listing accounts, displaying incoming images (through DM), and for creating stories from these incoming images. Furthermore, we updated the design of the home page.

The previous sprints were all created using html and css, due to some front and back end merging implications we were not able to completely setup a vuejs webpack environment. For the final sprint, however, we managed to do this. So all pages are somewhat redesigned and incorporated in a single page web application, meaning that the pages do not need to be refreshed each time a new web page is being displayed. Using the vuejs framework our web application consists of a number of components each presenting a different aspect of the web page view. The components 'Header.vue', 'Navbar.vue' and 'Footer.vue' are the only ones displayed continuously. Other than these the central part of the web page adjusts according to the router's specifications.

3.1.1 Header & Navbar

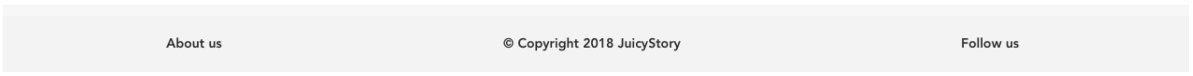
The header is displayed as a white rectangle at the top of the page, containing the juicy story logo, account name, home button, login/out button and the 'My Account' button. The design is simplistic and in tune with the rest of the web application's design. The juicy story logo links back to the web application's landing page, the home button links to the dashboard and the 'My Account' button will route the web app to the page where all Instagram accounts can be managed. Other than these, the login/out button is self explanatory.

The navigation bar is displayed below the header and each name holds the route to its respective component. For example, pressing the 'Incoming' button in the navbar will display the 'Incoming.vue' component in the central area of the web application.



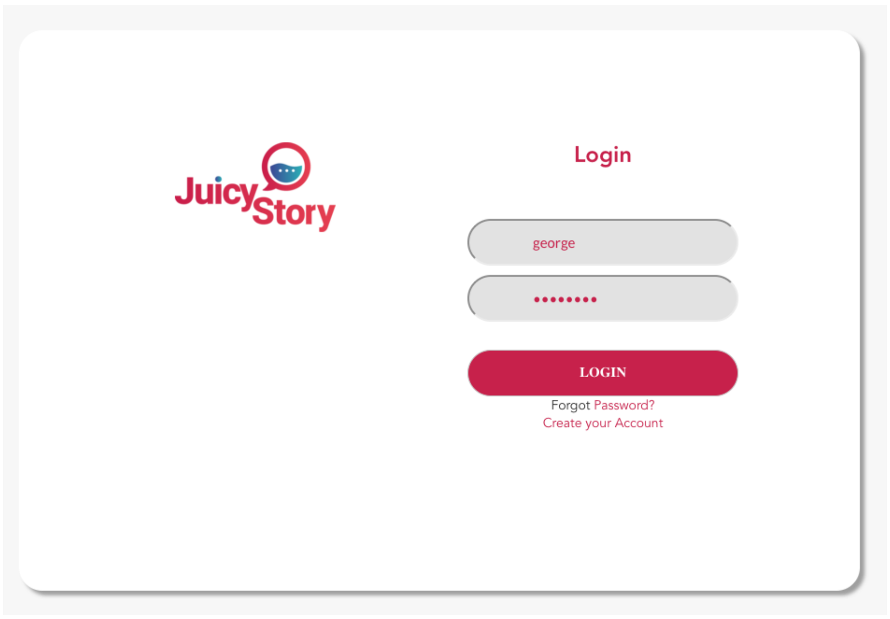
3.1.2 Footer

The footer is a grey rectangle located at the bottom of the page, containing an 'About us' and 'Follow us' section. Also, the copyright information is displayed here. Other than aesthetics and providing additional web page information there is not real functionality behind this component.



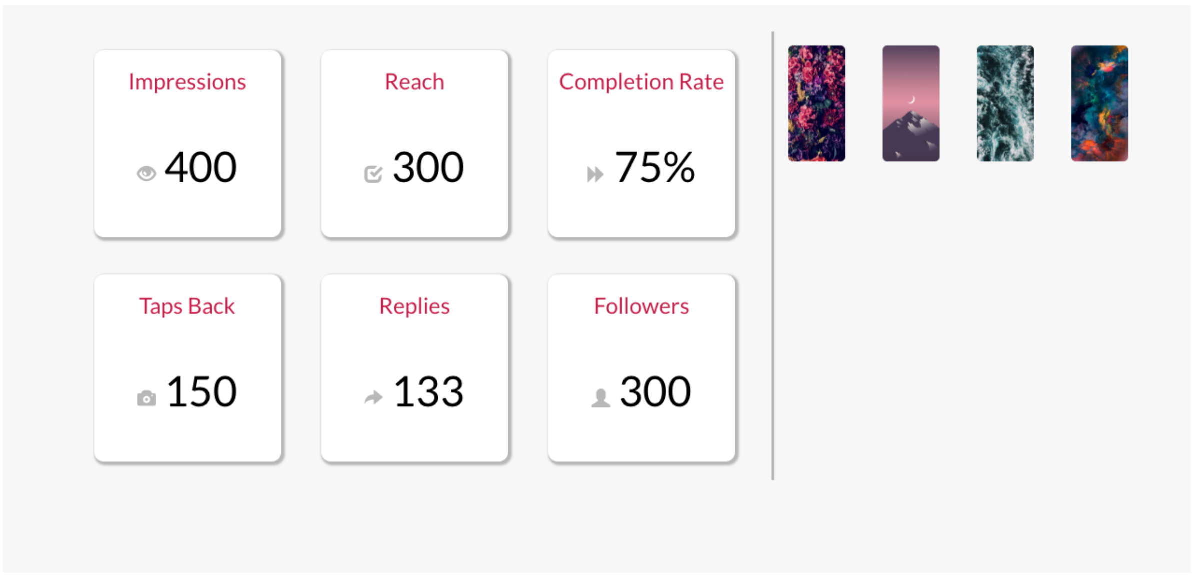
3.1.3 Login

The Login component consists of a generic login form which asks for the name and password of the account. The login button redirects the user to the home page of the application where they can find the entire content of their account.
As far as the design goes, we created a minimalistic interface focused on the main element: the login, with the juicy story logo located to the left of the form.



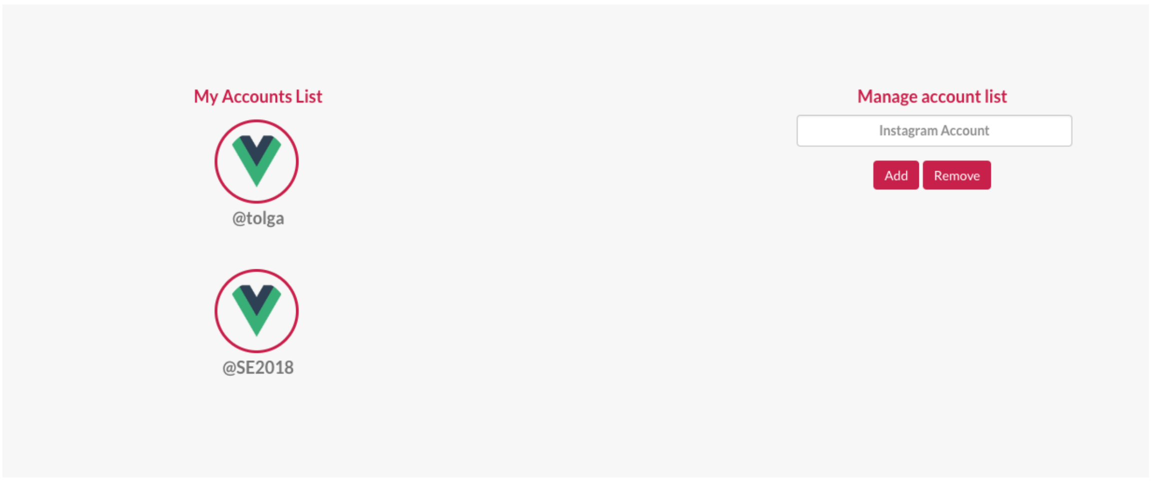
3.1.4 Dashboard

This component gives an overview of the story of the selected Instagram account. This overview is given in a grid of cards, and next to this grid some incoming stories are already being displayed. No functionality is added to this component besides the presenting of this data.



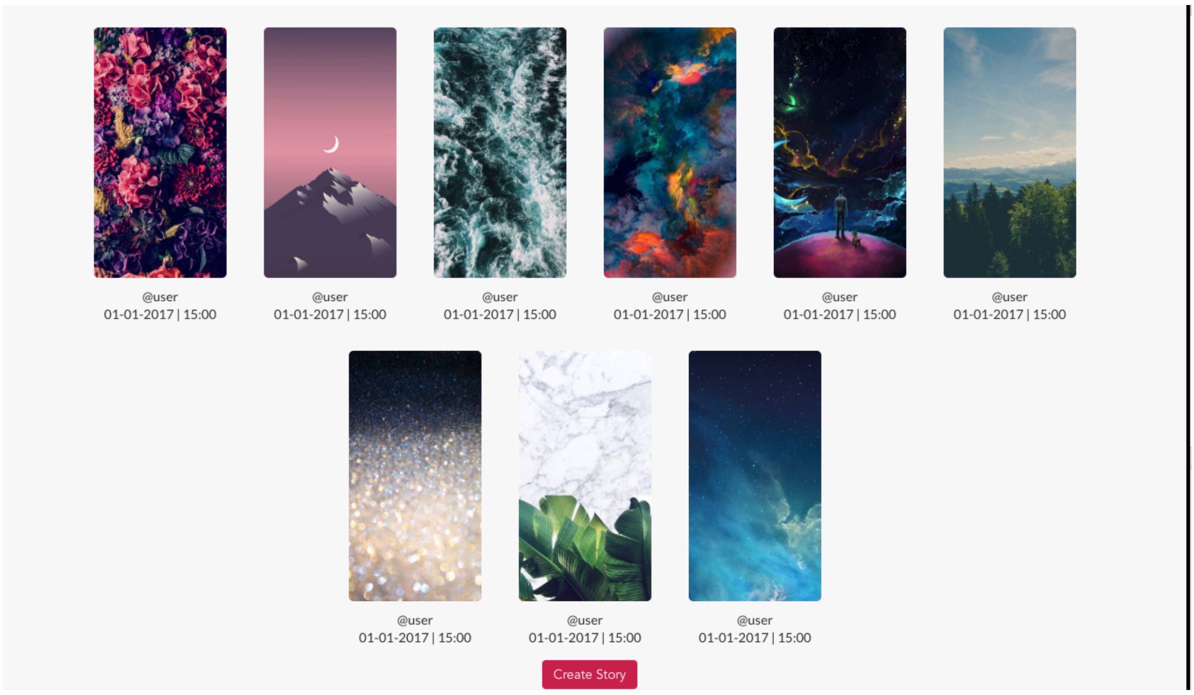
3.1.5 Account list

On this page, the user can view a list of the Instagram accounts linked to his/her Juicy Story account. The user can add or delete any account, and select accounts to be managed from the Juicy Story web app.



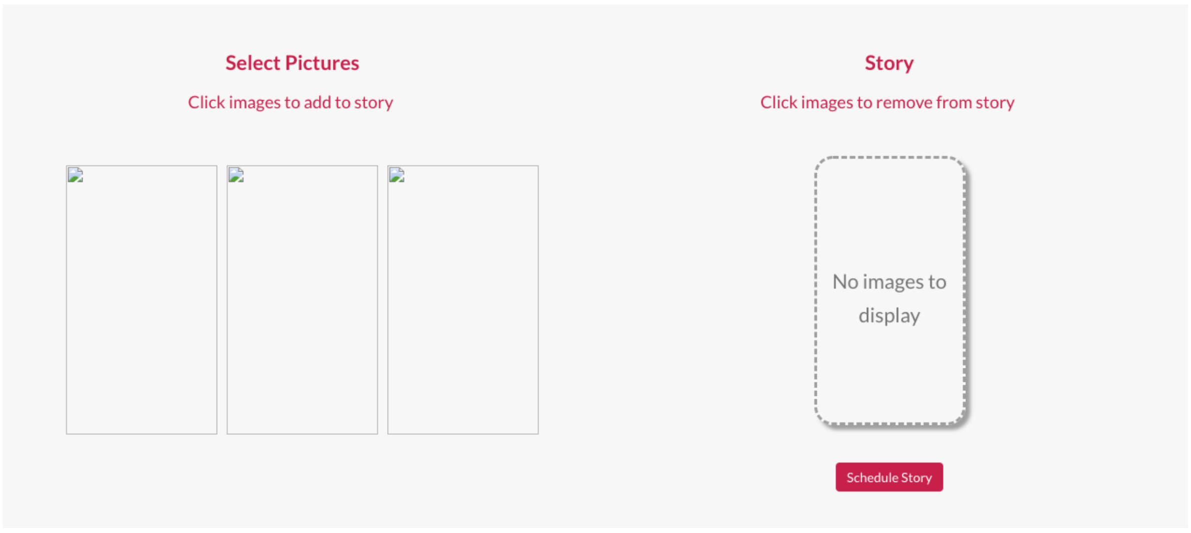
3.1.6 Incoming

The incoming tab displays all images that were sent to the currently selected Instagram account via direct message. The user has a possibility to select these images. When the user is done selecting images, the 'Create Story' button can be clicked. The user is then sent to the Story Creator tab, where a story with the selected images can be created.



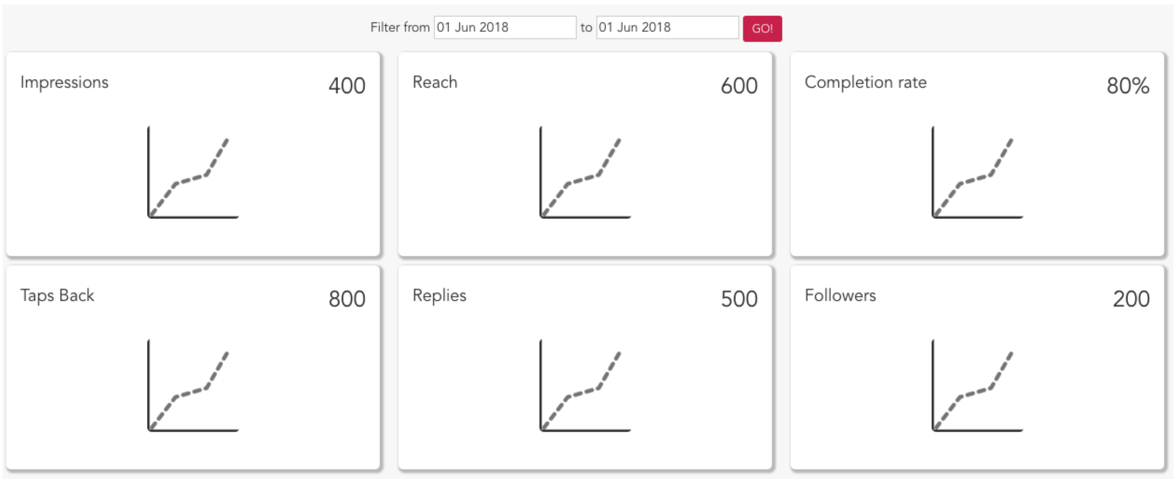
3.1.7 Story Creator

On this page, the user is able to create stories using the images selected on the incoming tab. On the left, these selected pictures are listed. By clicking on these pictures, they are pushed to the right of the screen. This is where the story is created. The images appear in the order on which you clicked on them. When the user has added all images to the story in the preferred order, the 'Schedule Story' button can be clicked. This button allows the user to specify the time and date to upload the story to the Instagram account.


Select Pictures
Click images to add to story
Story
Click images to remove from story
No images to display
Schedule Story

3.1.8 Report

This component comprehensively presents the statistical information with regards to the current account's stories. The page interacts with the back end to obtain this data and displays it in a grid-like formation. The filter at the top of this component allows the user to specify a specific range from which the data should be displayed. The spaces where the date can be filled in provide a small calender to select a date in order to increase user friendliness.



Filter from 01 Jun 2018 to 01 Jun 2018

Impressions 400

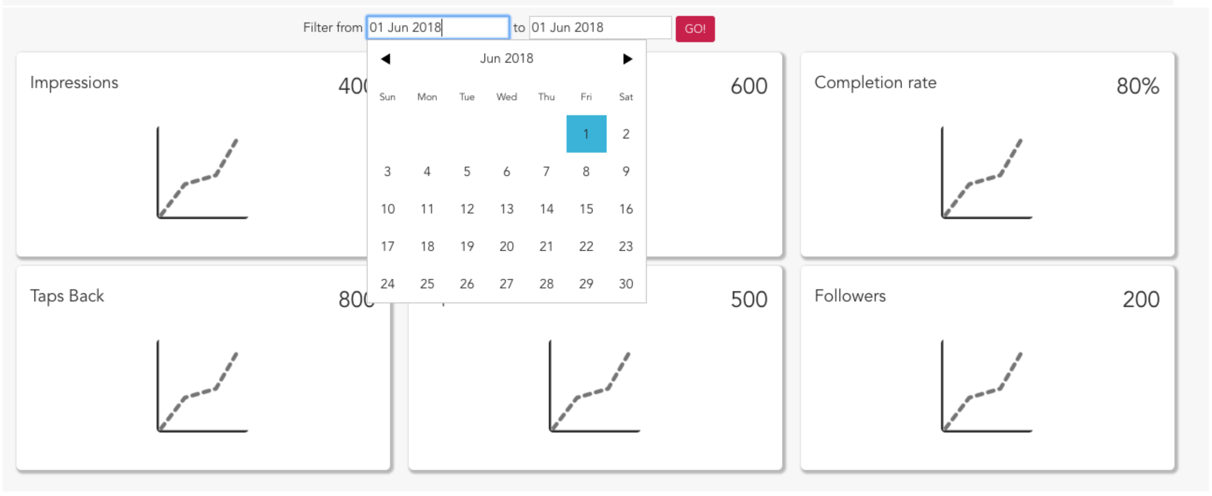
Reach 600

Completion rate 80%

Taps Back 800

Replies 500

Followers 200



Filter from 01 Jun 2018 to 01 Jun 2018

Impressions 400

600

Completion rate 80%

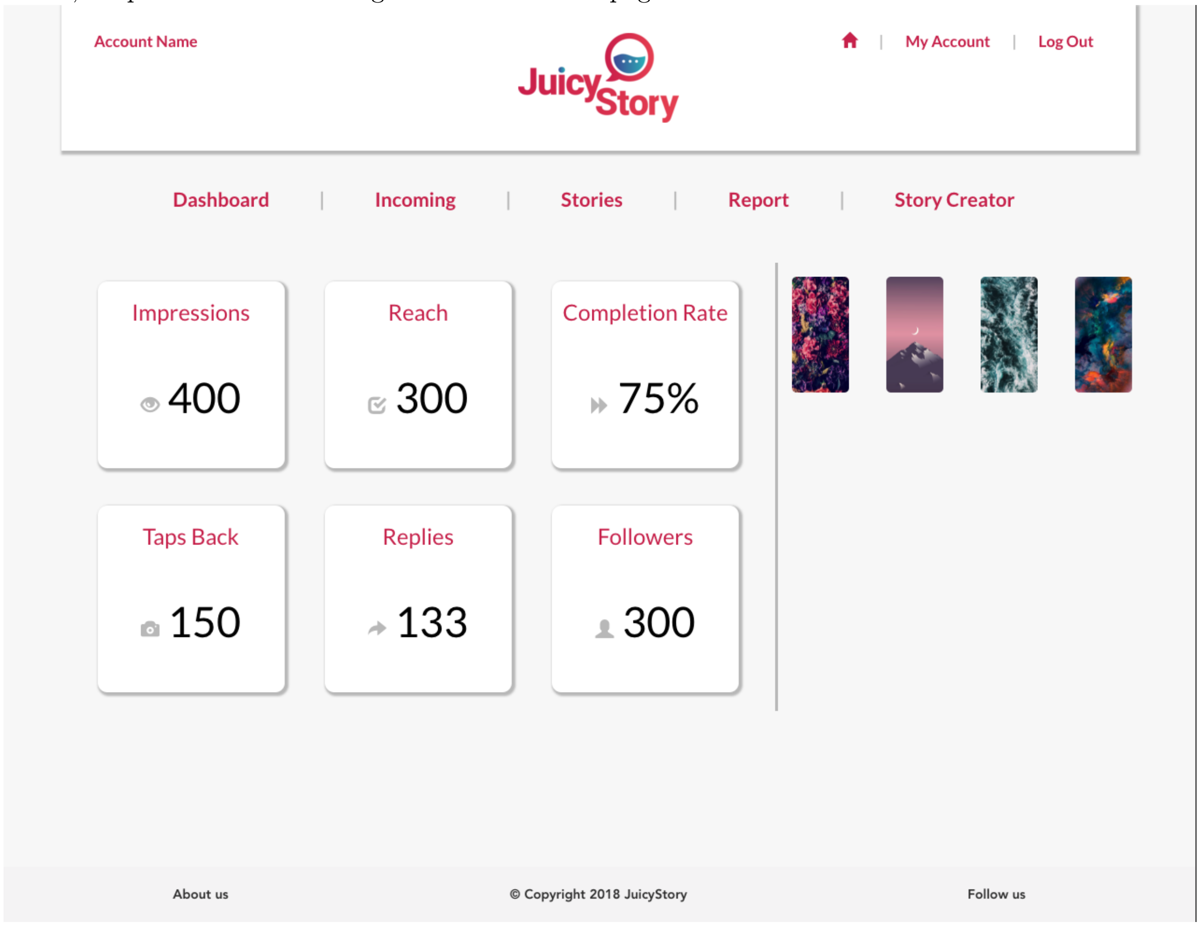
Taps Back 800

500

Followers 200

3.1.9 Combined overview

In order to clarify the idea what the header, navbar and footer together with the central component looks like, we present the total image of the dashboard page below.



Impressions 400

Reach 300

Completion Rate 75%

Taps Back 150

Replies 133

Followers 300

4 Team Organization

Team organization has been a large influence on how this web application came to be. The usage of several communication tools has benefited the efficiency tremendously. For the most part we divided the team in a front end and back end sections, each member spend the majority of his/her time working on problems related to their respective section. Also, full-stack members were present, who made sure the merge between front and back end went smoothly. Below an overview is given of the main instruments used to achieve this.

4.1 Meetings

Weekly and at times twice weekly meetings made sure all team members were on the same page and up-to-date with the latest developments. Also, tasks could be divided during these meetings and difficulty with certain problems could be discussed.

4.2 Trello

To keep track of all 'tasks performed', 'todos' and 'in progress' we used Trello. Each team member had access to the information and was able to alter it accordingly.

4.3 Slack

All team communication was performed using Slack, a communication app with options to keep several threads. This separation of discussions allowed for all communications to stay organized.

5 Change log

Table 1: Change log

When	**What**	**Who**
23/02/2018	Created basic overview of application for back end and front end.	...
12/03/2018		...
16/04/2018		...
3/06/2018	Updated all pictures to correspond with new appearance. Added missing front end page explanations and updated old ones. Added team organization section. Added change log. Added content page. Added introduction.	Casper