Software Engineering Product Architecture Version 2

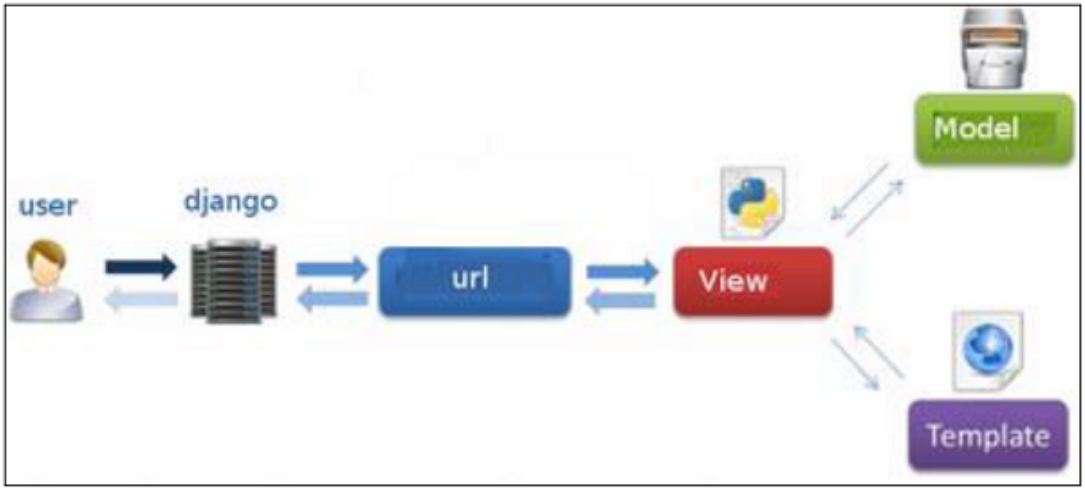
Juicy Story Project Group
Hichem Bouakaz (TA)

Tolga Parlan S3051072
George Argyrousis S3175243
Alina Matei S3080722
Casper Schmit S3194299
Michiel de Jong S2550768
George Rakshiev S3015564
Wim de Jager S3215210

May 9, 2018

Overview

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.



Back-End

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 Instagram user info such as the access token and user name.

- 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.

- iusers : The application responsible for Instagram account management. The action of adding an Instagram account and showing the list of all Instagram accounts managed by the user are handled by this application. The model is comprised of the Instagram handle and a foreign key to the user that it is linked to.

- 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.

Front-End

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

1. HTML & CSS
2. Bootstrap
3. Vue.js (framework)
4. Javascript (jQuery)

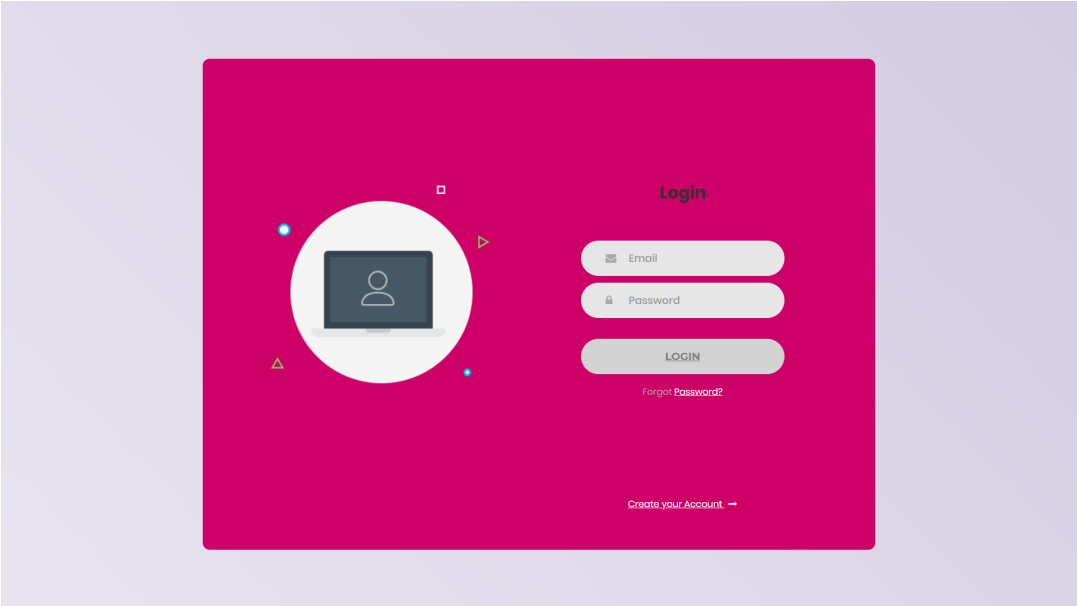
Architectural and design decisions

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.

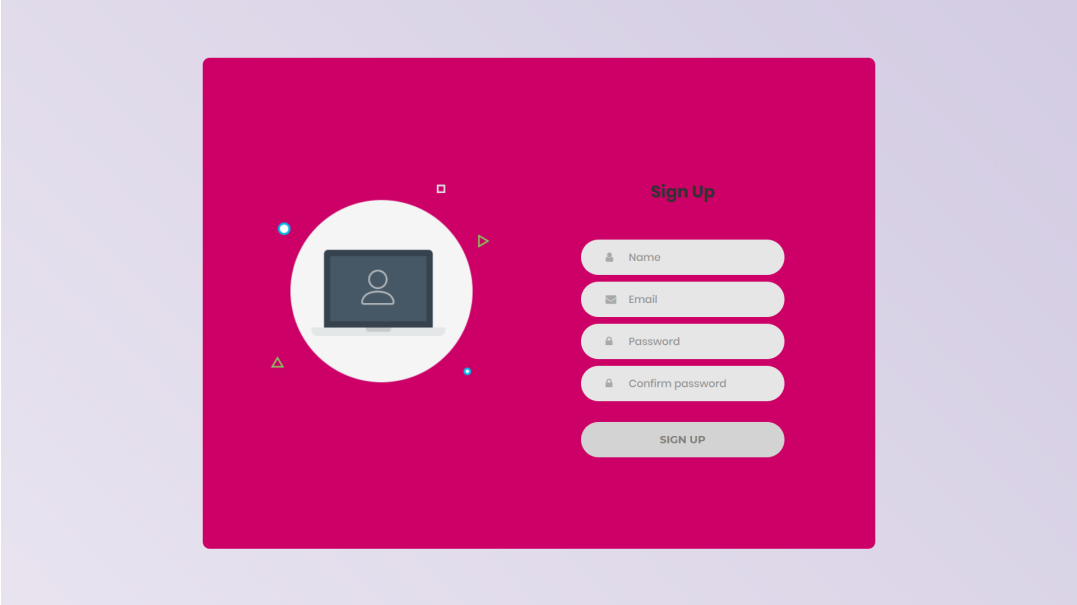
Login

The Login page consists of a generic login form which asks for the email address of the account and respectively, the password. The login button redirects the user to the home page of the application where they can find the entire content of their account.
We added 2 basic extra options: creating a new account and password recovery. The links redirect the user accordingly.
As far as the design goes, we created a minimalistic interface focused on the main element: the login. Right to the login fields, the logo of the company will be added to personalize the page.



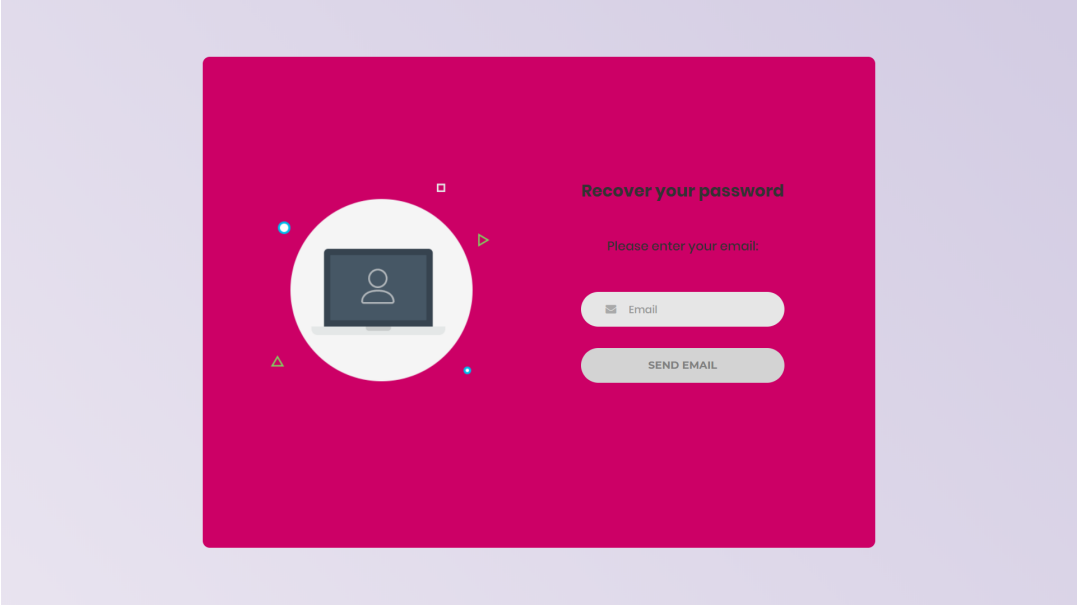
Sign up

The main focus of the Sign up page is on the form with the following fields: name, email address, password, password confirmation. All the above mentioned fields are required in order to create the account together with a valid email address (xyz@wt). Another requirement is having the content of the two password fields to match.
The design is similar to the Login page in order to maintain consistency.



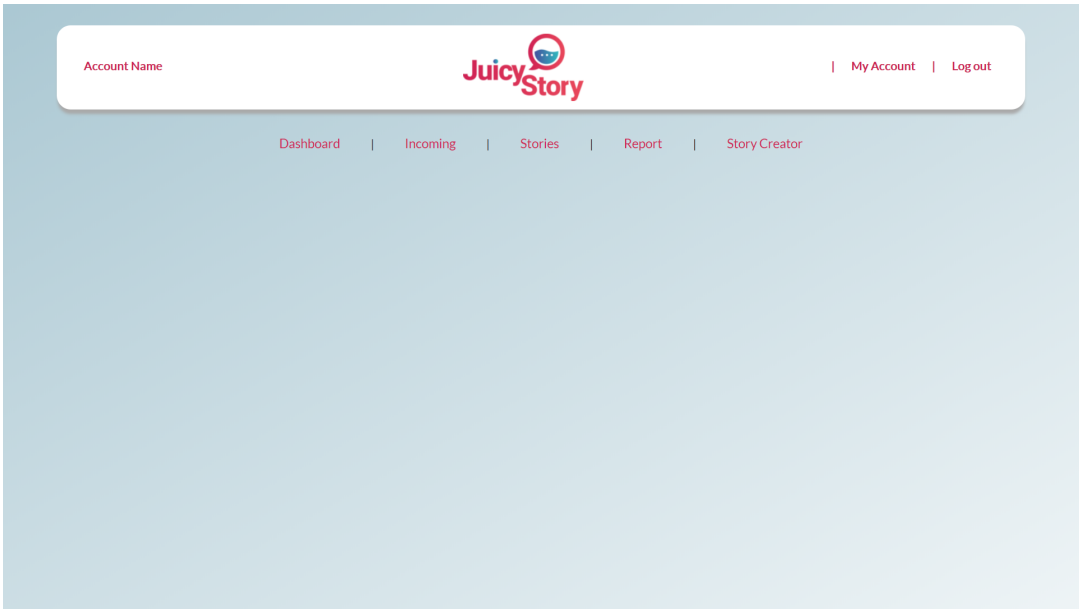
Password recovery

The Password recovery page only consists of one field requiring an email address in order to request a password change.



Home

Every page contains a title bar and a menu with several tabs you can navigate through. The home page contains just these elements. From this page, we will create the other pages.

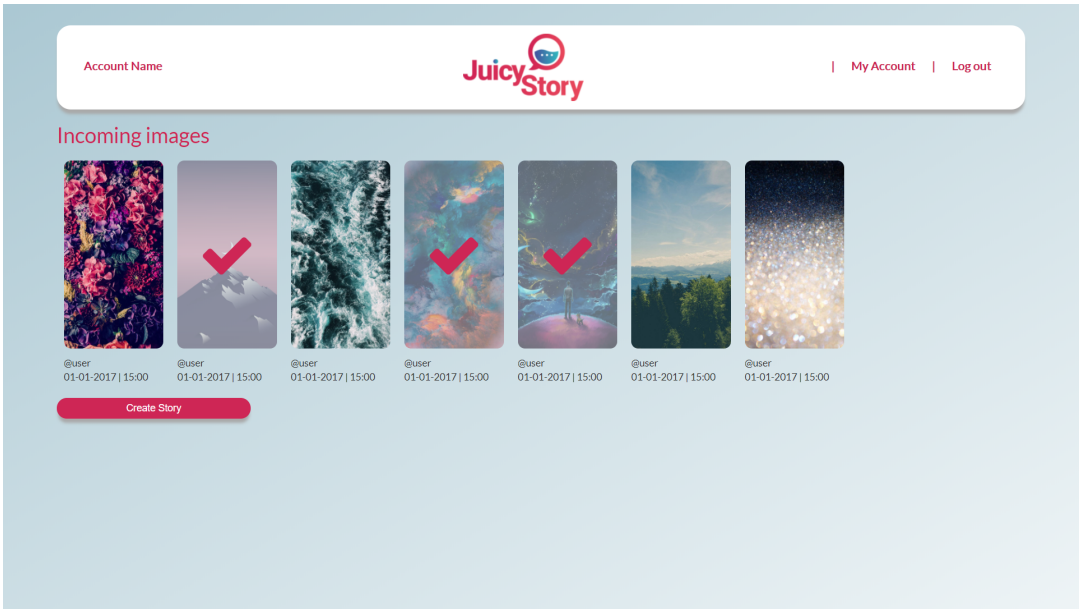


Account list

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

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 he is done selecting images, he can click on the 'Create Story' button. The user is then sent to the Story Creator tab, where he can create a story with the selected images.



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, you push them to 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 added all images to the story in the preferred order, he can click on the 'Schedule Story' button. Although not yet implemented, this button allows the user to specify the time and date to upload the story to the Instagram account.