**High, Travel!**

**Final Report**

June 15, 2015

**Members**

Team 9 김지우, 박규연, 양지훈, 최현지

**Project Abstract**

"High, Travel!" will provide a service for people planning trip with someone else. Our service can be of help in all processes of trip, from start to end. While planning the trip, users share the itinerary and budget. Also they can confirm travel routes with map and come up with the rules. During the trip, users can record their expenses, organize pictures by date and places. Using these photos, users can write a diary privately. Finally, after the travel, users can solve the financial problems easily, and look back on their trip with photos and diary neatly organized.

**Motivation**

What are you planning to do in this summer vacation? Many people want to travel with their friends in their vacation because they can refresh themselves in the trip. However, to have a memorable trip, travelers have to consider many things like schedules, expenses, and photos. Especially when they travel with someone else, tiny things can make their trip worse. Therefore, we decided to develop the web service which syntagmatically manage the travel. By using our service, travelers can easily handle common issues like schedules, places to visit, todos, budget, and expenses. Moreover, they can specify rules to set boundaries and share photos in organized way. For individual users, we provide private diary menu to remember their trips longtime. Our service will improve trip experiences from the start to the end.

**Related Work**

* Individual travel planning sites

These sites like "Tripit" provide booking or budget management for individual users. To use these sites, users have to register to many different sites to handle the whole trip. Even worse, trip has much more things to consider and it is cumbersome to find all sites for them. Moreover, according to 2016 statistic from Korea Tourism Organization, only 12.5% of Koreans went on a trip alone, and average number of people traveling together was 5.7. This result shows that there might be a lot of needs of sharing their whole traveling process between members. Thus, our service provides add user to trip function and each user can record their own expenses and share photos on our service.

* Travel sharing site

"Travefy" service have some common points with the service we are planning to develop. This service supports travel planning with maps and sharing. However, its target customer is travel agent, and agents need to pay for the service. Compared to this service, our service target on every traveler, and more focused on the meaning of traveling together. We also show schedules in calendar for planning, and use map to mark places to visit.

Additionally, we provide service like photo sharing, budget and expense log, and posting rules. Traveling together means making meaningful memories with traveling members. By sharing their photos, users can make their trip more impressive. Moreover, by partitioning shared photos’ page by date, user can manage their photos effectively. Keeping track of budget and expense can facilitate splitting money between users, which can be very sensitive and important problem. Posting rules function will help the members to avoid conflicts during their trip. Finally, our service also provides diary function. Users can scrap photos to their diary and organize their travel memories individually.

**Functionality**

* User Registration & Sign in

New users can sign up simply in signup page with username and password. Then, users can log in by their username and password in intro page.

* Manage Trip List

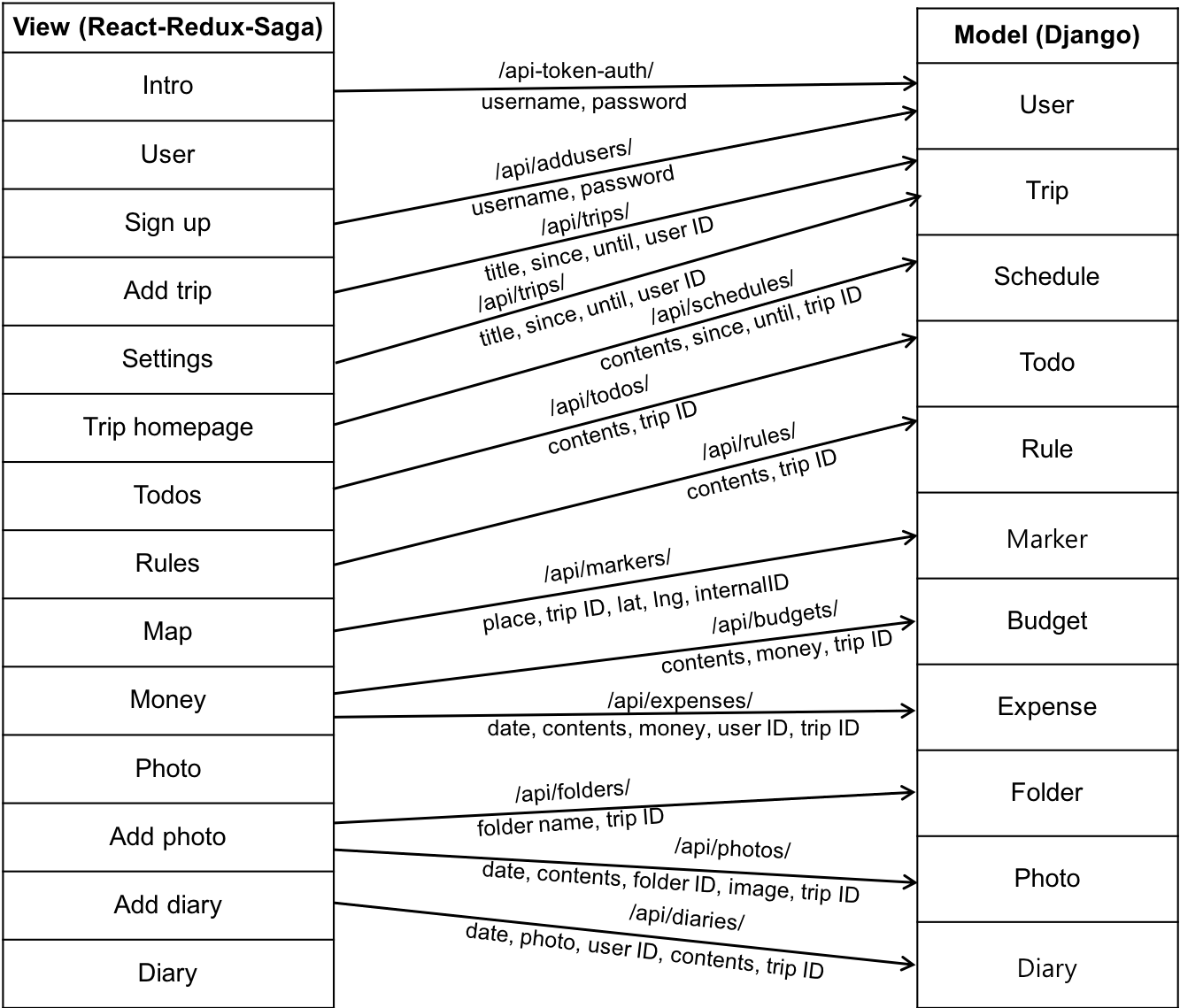
User page shows user’s trip titles with duration. Users can create new trips and delete trip if he/she is the creator.

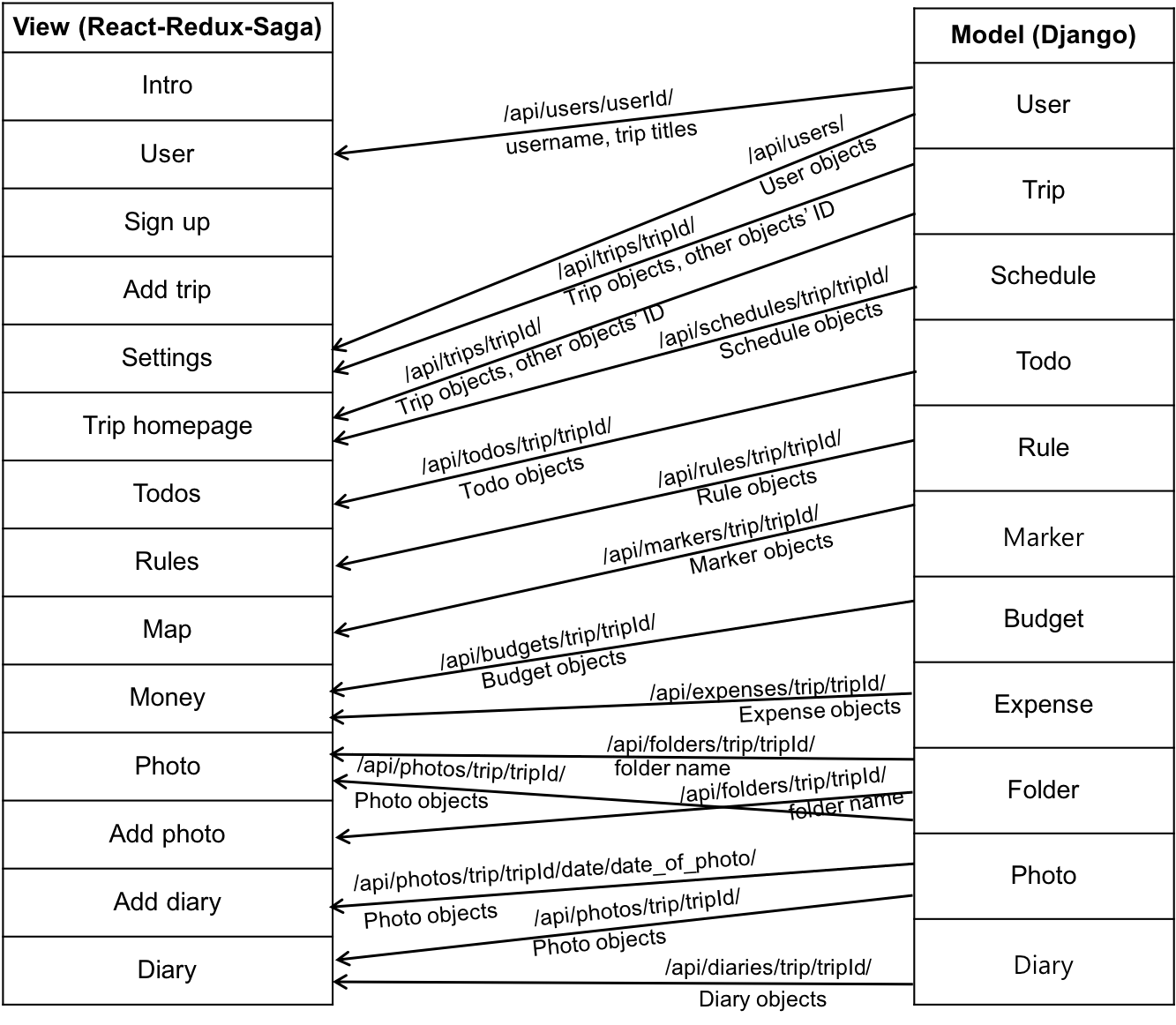
* Manage Trip Details

In home menu, users can check their schedules in calendar and schedule list. They can add or delete schedules. In rules and todos menu, users can see rules to follow, and add or delete rules. They can also make todolist and mark done or add or delete todos. In money menu, users can budget and record expenses and get total budget cost, total expenses for each user calculated. Users and add budget or expense and delete budget or his/her own expense. In map menu, users can mark places to visit in google map. Then users can add place name and ID to markers and delete markers. When they click go button of markers they can see that place in the map. In photo menu, users can see photos under folders and download or delete photos by selecting individual photos or folders. They can add folders and multiple photos at once in add tab. In diary menu, users can see diary and modify contents or delete diaries. They can write a new diary with photos that they uploaded. Finally in settings menu, users can see trip title, duration, and accompanies. Only the creator of the trip can change trip details and user list by add or delete.

**Design & Implementation**

**MVC**

****

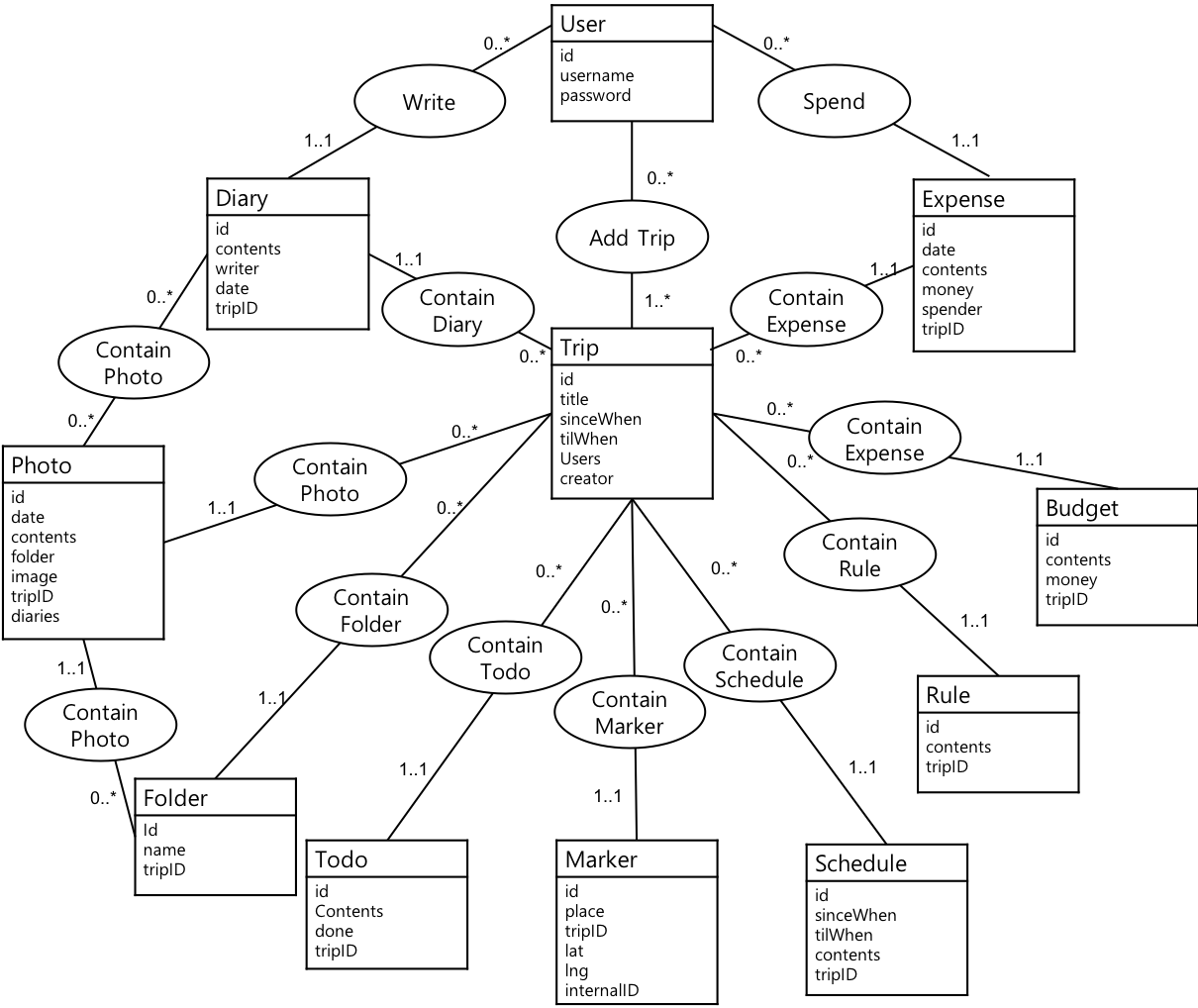


There are 14 views (left) and 11 models (right). For readability, budget tab and expense tab were integrated to money views. Arrows show controllers, left-to-right showing requests with user inputs from the view, and right-to-left showing responses from the models with data. Above the arrows, the backend APIs that frontend views will use to communicate with models are shown.

**Backend**

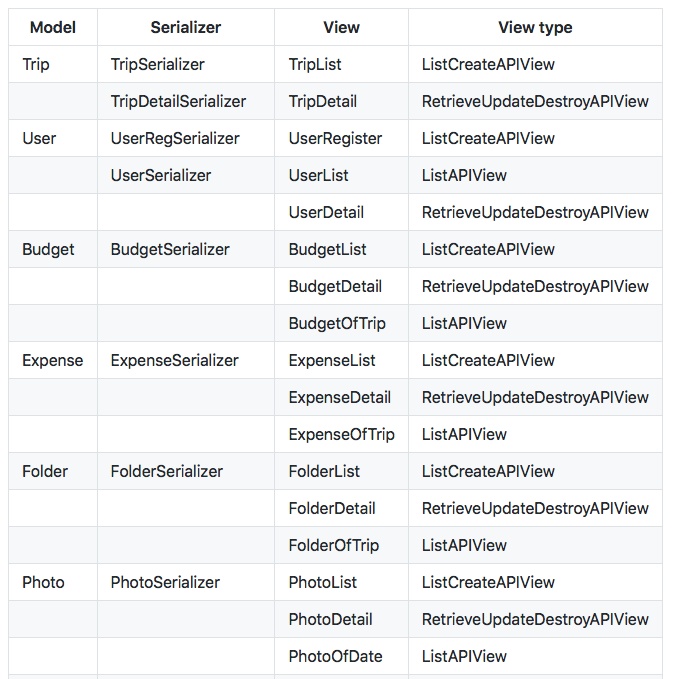
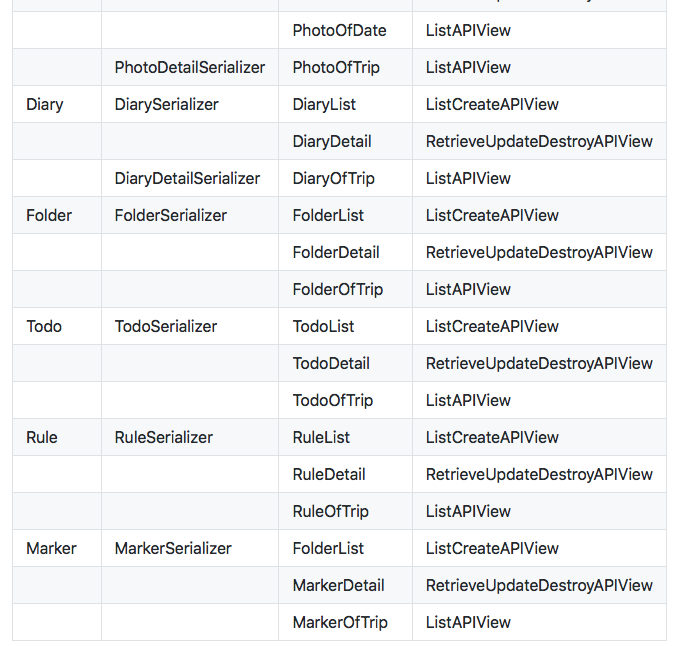
**Model Details**

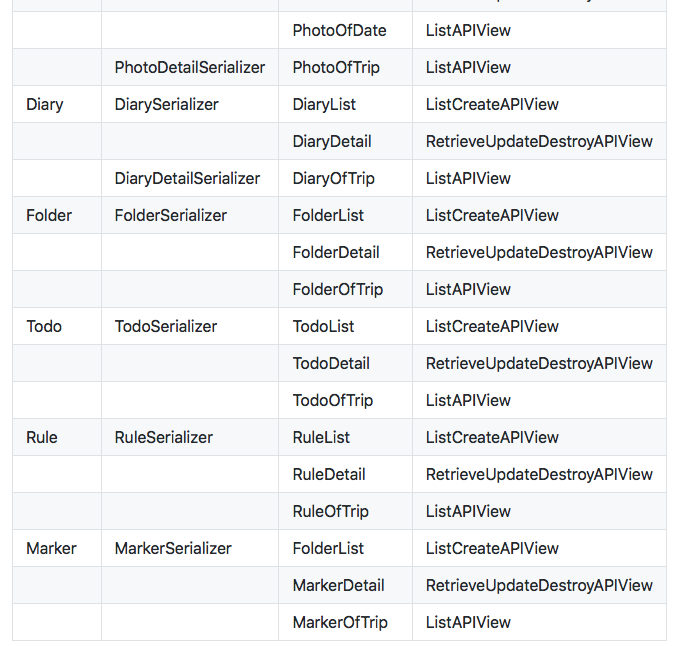
Here is an Entity-Relationship diagram for model design.



Each rectangle represents an entity with attributes listed inside. Each oval represents the relationship between entities, numbers next to the line showing cardinality constraints (min..max). In this relationship, cardinality constraints combination of 1..1 and 0..\* represents one to many relationship. On the other hand, 0..\* and 0..\* combination(photo and diary relationship), 0..\* and 1..\* combination(user and trip relationship) represents many to many relationship.

**Models with Backend Views**





Basically, every newly defined model has a serializer with all the fields it has, including related name of foreign keys and many to many fields. Moreover, each has ListCreateAPIView to make new instances, and RetreiveUpdateDestroyAPIView to modify fields or delete certain model instance. For frontend implementation convenience, ListAPIView is added for each model. This view shows all model instances of certain trip.

Some model has two different serializer according to its usage. For example, TripSerializer shows all foreign keys with their own serializer, but TripDetailSerializer shows only their IDs. Furthermore, PhotoSerializer and DiarySerializer each shows folder and photo field with IDs, but PhotoDetailSerializer and DiaryDetailSeralizer shows FolderSerializer and PhotoSerializer respectively. This changes facilitate rendering photo and diary pages in the frontend.

Finally, user model was imported from django.contrib.auth.user. We added new method with post\_save signal to allocate token for each user instance. For sign up, we made UserRegSerializer with username and password fields only.

**Libraries**

- django-cors-headers : Django application for handling the server headers required for Cross-Origin Resource Sharing.

- drf-writable-nested : writable nested model serializer for Django REST Framework, facilitate handling many to many fields.

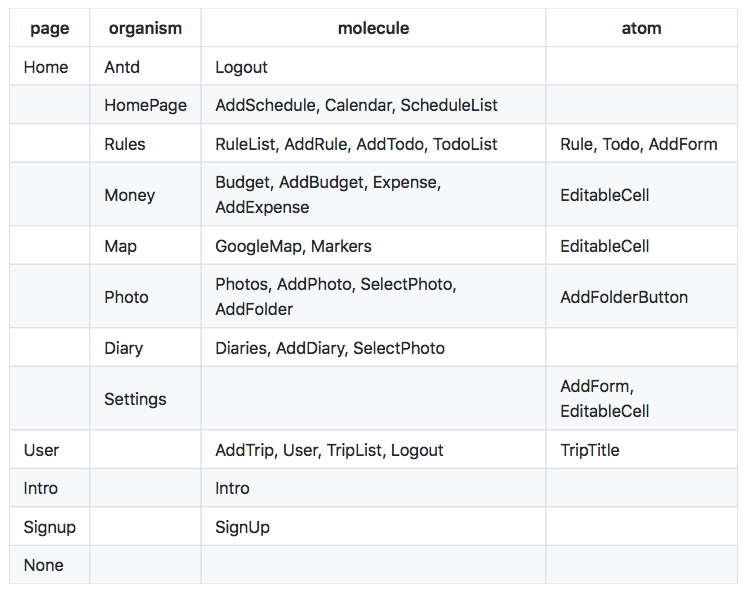
**Testing**

- django.test : testing tools with client (dummy Web Browser), and TestCase class provided by Django. We tested all views and models. (Unit test, Functional test)

- coverage : code coverage measurement for Python. We confirmed that all lines were covered.

**Frontend**

**React Components**

****

Our service consists of 5 pages. Each page has different URL by react-router. None page is for all invalid URLs, showing 404 error code with redirect button to Intro, Signup, User page. Intro page has Intro molecule that shows service title, username and password inputs, login and signup buttons. User page has User molecule with username and Logout molecule, and TripList, AddTrip molecule for handling trip list. TripList's titles are TripTitle atoms. Home page has 8 organisms that indicates each menu and overall layout. Antd organism get other organisms and show them in menu with header and sider. Homepage organism supports scheduling, and Rules organism supports rules and todos. Here, each rules and todos are treated as atoms. Next, money organism uses Budget, AddBudget, Expense, AddExpense molecule with EditableCell atoms to render budget and expense table. Photo organism consists of Photos molecule that show photos under folder name, SelectPhoto molecule to select some of the photos, AddPhoto molecule that uses Antd form to get user inputs, AddFolder molecule under AddPhoto molecule as one of the options. Diary organism also uses SelectPhoto organism to pick photos to use in diary. Finally, Settings organism uses Antd table with AddForm and EditableCell to support trip detail management.

**Libraries**

- antd : overall frontend user interface design.

- react-router v3 : frontend routing.

- fullcalendar : JavaScript event calendar.

- fullcalendar-reactwrapper : react wrapper component for the fullcalendar.

- react-photo-gallery : show photos with customized size and support selection.

- axios : Promise based HTTP client for the browser and node.js. We used axios to send formdata with photos in Saga.

- google-maps-react : including a helper to wrap around Google maps API.

**Testing**

- jest : Jest is used to test JavaScript code including React applications. Jest was also used to check our frontend code coverage test.

**User Interface**

**Conclusion**

Our interests on travel have grown bigger and bigger as our income increased and we want to relax our stress from jobs. We are sure that travelers who use our service will have a good trip. There are few web services related to travel, but most of them miss some important things. Only our service can fully manage the travel from the start to the end. We provide a variety of services like scheduling, Google map API, and photo sharing.

To accomplish this, we used django, restframework for our application's backend and react, redux, saga for frontend. we use django restframework in backend. In frontend, we used many react libraries. For example, we used ‘fullcalendar’ in schedule menu and ‘react-photo-gallery’ in photo page. To apply attractive design, we used ‘antd’ library.

For the future, we might need to get fund for the photo storage. Then we will be able to handle hundreds of photos for each trips. Also, we can add currency converter in money page to facilitate expense logging. Moreover, we might want to improve security problem in sign up process, with e-mail confirmation. Finally, we can add invitation mail system for adding users to trip. Still, we are sure that you will have complete travel management by simply accessing website.