Project 3: Tripper

Design Analysis

Ryan Lacey, Jesika Haria and Ido Efrati

6.170 Fall 2013

October 20, 2013

1. Overview:

[Primary author: Jesika Haria]

Tripper is a browser-based cost sharing application that was designed to help groups of friends going to a road trip to upload and share their costs. The uniqueness of Tripper is that it allows users to submit both monetary and non-monetary (e.g. driving time) costs. Users can create and invite friends to a road trip, upload and share their costs, and to split costs within a group seamlessly and effortlessly.

2. Purpose and goals:

[Primary author: Jesika Haria]

Road trips with friends are fun, but splitting the costs is not because:

a) People often forget how they incurred the cost unless they record it immediately
b) It is hard to visualize the total amount of money spent on the trip
c) Costs are not all monetary: driving the car, planning, etc. all need time and effort, which should count towards a common cost
d) It is hard to figure out who owes who at the end of the trip, especially when there are more than two participants.

We're developing Tripper so that you can have fun on a road trip with friends without the inconvenience of splitting costs. With Tripper, you can:

a) Record your expenses as you go
b) View and analyze your expenses and compare it to your friends in a clean interface
c) Our unique way of defining cost in terms of 'tokens' allows you to set an exchange rate for the value you bring to the trip. For example, 1 USD = 10 Tokens; 1 hour of driving = 5 Tokens.
d) Our unique splitting tool aggregates everyone's costs (in token units) for a trip, converts it back to USD, and allows you to pay to a common pool that is split up across outstanding recipients.

3. <u>Motivation and Context:</u>

[Primary author: Jesika Haria]

We chose this application specifically because we're avid road trippers and campers, but find applications like Billmonk to be essentially fancy calculators. Also, there are no applications in the market right now that allow us to define our own currency exchange for activities, and we want to factor that in making a truly equitable split for road trip costs.

This is the real power of Tripper: to allow users to collectively value their friends' effort and time in making their road trips awesome. Tripper is positioned to take advantage of both the niche market of road trippers, and introduce an innovative model for 'cost'.

4. <u>Minimal Viable Product:</u>

[Primary author: Jesika Haria]

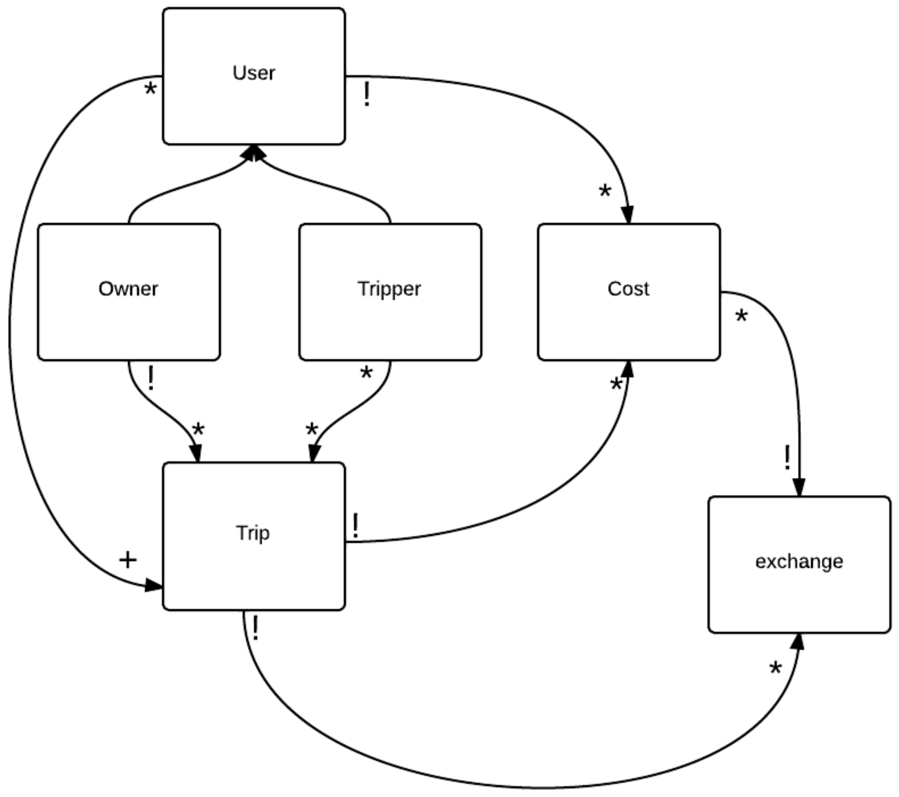
A minimum viable product would be:

a) **<u>Before the trip:</u>** A user should be able to create a trip and define a group of existing users as eligible to submit costs to the trip.
b) **<u>During the trip:</u>** Every user should be able to view a trip's total costs, their own share of the costs, and add or delete their costs from the trip. The minimal definition of cost is monetary cost, measured in USD.
c) **<u>After the trip:</u>** Each user should be able to cash out or repay their remaining balance.

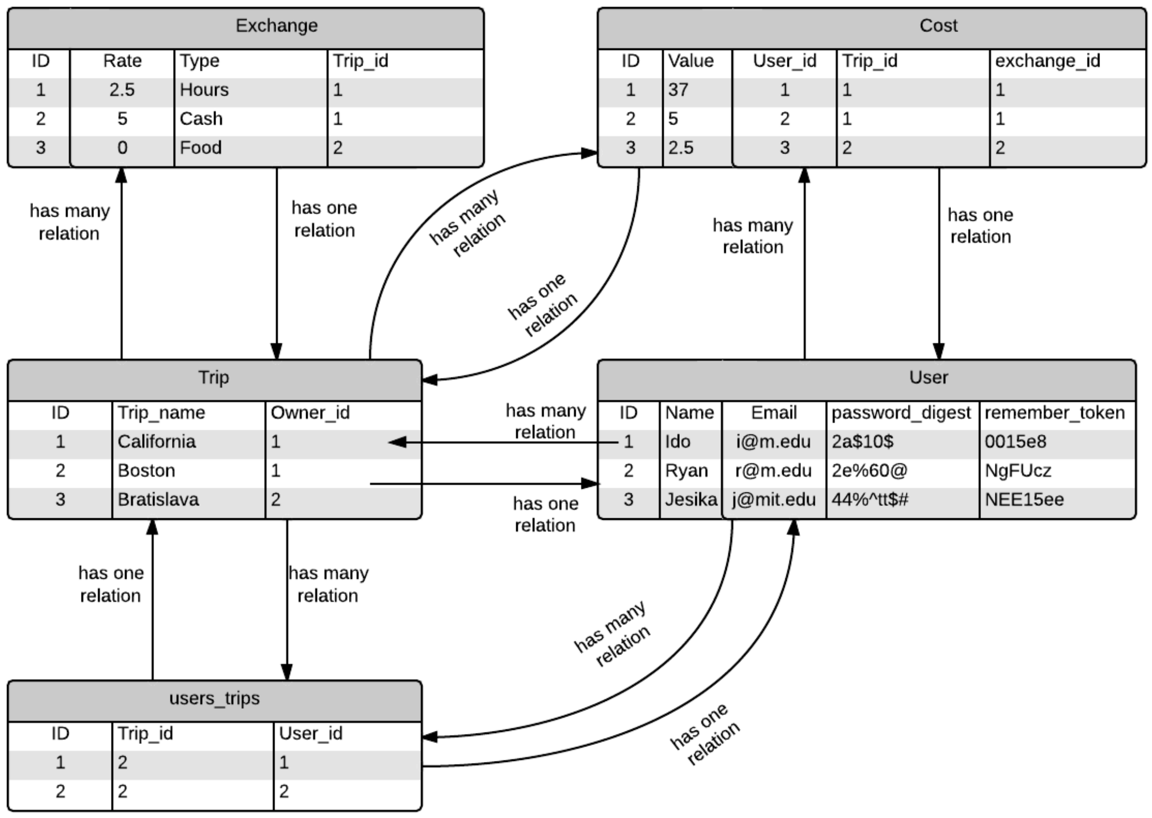
5. Concepts

[Primary author: Ido Efrati]

5.1. Data Modal



5.2. <u>Database design and relations</u>



Exchange

ID	Rate	Type	Trip_id
1	2.5	Hours	1
2	5	Cash	1
3	0	Food	2

Cost

ID	Value	User_id	Trip_id	exchange_id
1	37	1	1	1
2	5	2	1	1
3	2.5	3	2	2

Trip

ID	Trip_name	Owner_id
1	California	1
2	Boston	1
3	Bratislava	2

User

ID	Name	Email	password_digest	remember_token
1	Ido	i@m.edu	2a10	0015e8
2	Ryan	r@m.edu	2e%60@	NgFUcz
3	Jesika	j@mit.edu	44%^tt$#	NEE15ee

users_trips

ID	Trip_id	User_id
1	2	1
2	2	2

As seen in the above diagram Tripper consists of five tables, a trip table, a user table, a cost table, an exchange table and a `users_trip` table.
The user table has an encrypted password to protect from attacks, and an encrypted session token to prevent from sessions attacks. Most of the validation is performed only at the model level (presence of required field, and correct format), but uniqueness validation is performed both on the model level and on the database level. The reason behind the double uniqueness validation is to prevent a duplicated creation of an entry, if the user clicks the sign up button twice with a slow Internet connection.

A Trip has a title, and an owner (the user who created the trip). A trip owner cannot be changed after a trip was created. The owner field stored as an integer that corresponded to the primary key of a user in the user table.

A `users_trips` <u>table</u> has a `trip_id` stored as an integer that corresponded to the primary key of a trip in the trip table, and a `user_id` (the user who has access to that trip) stored as an integer that corresponded to the primary key of a user in the user table.

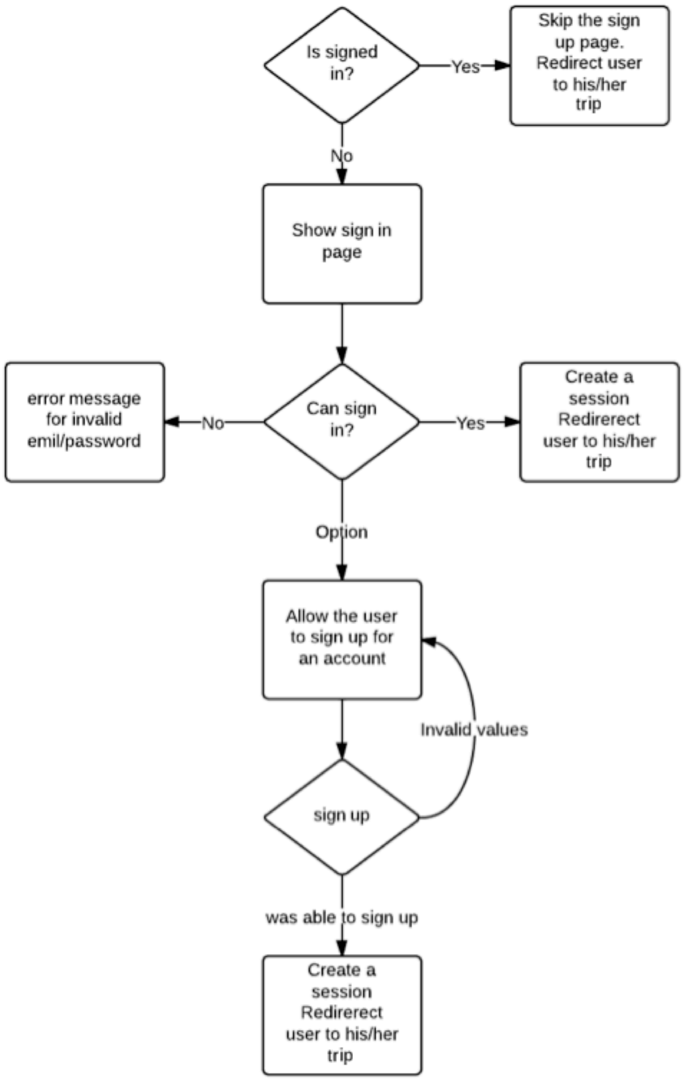
An exchange has `trip_id` stored as an integer that corresponded to the primary key of the trip in the trip table. Each trip has its own exchange <u>rate</u>.

Finally, a cost has a `trip_id` stored as an integer that corresponds to the primary key of the trip in the trip table, a `user_id` stored as an integer that corresponds to the primary key of the user table, and an `exchange_id` stored as an integer that corresponds to the primary key of the exchange rate.

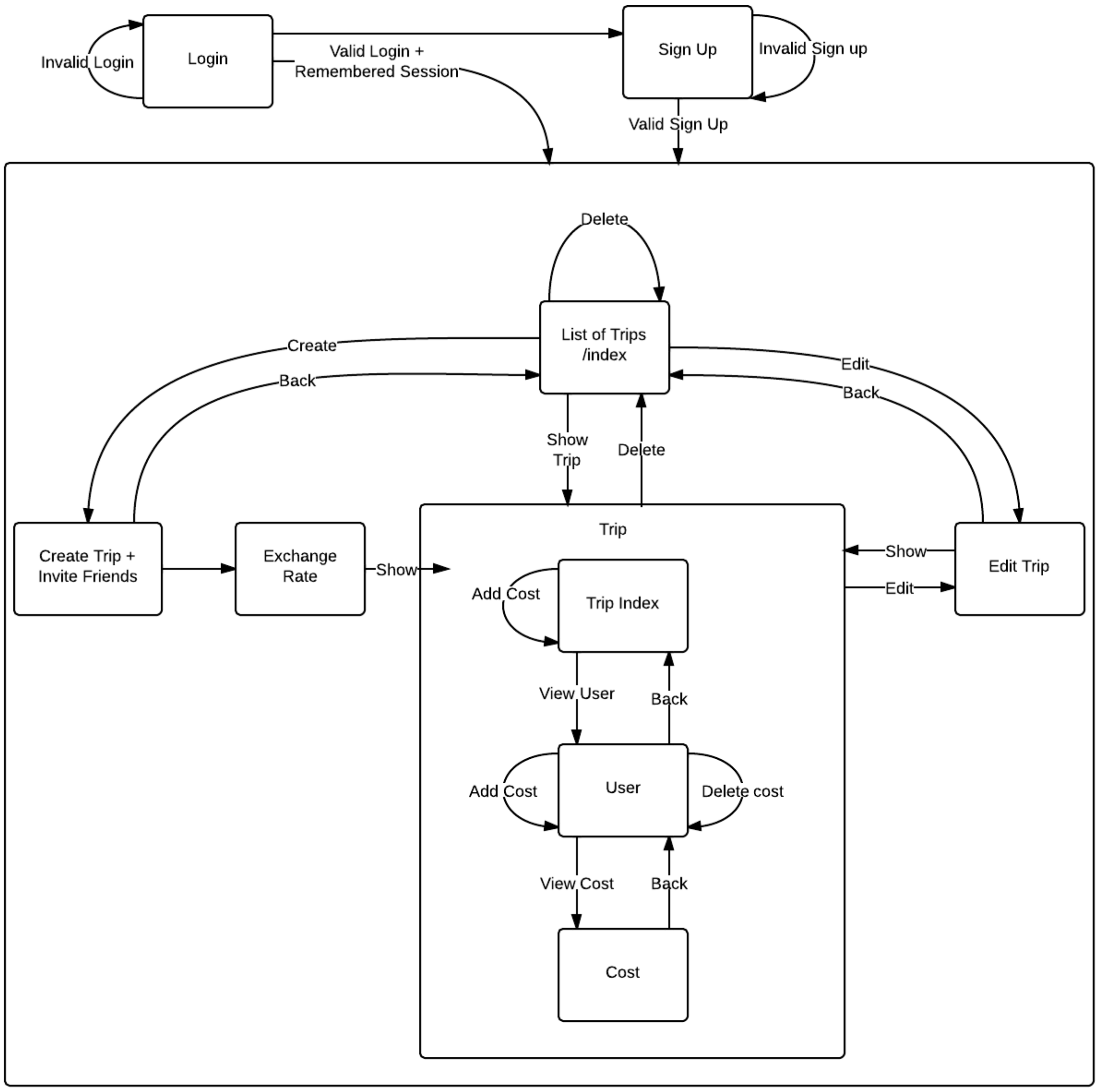
5.2.1. Dependencies

There are several dependencies in the above scheme. If a user was deleted from the database all of their costs will be deleted as well. If an owner deleted a trip from the database, all of the costs associated with this trip, and the `users_trips` relations associated with this trip will be deleted as well.

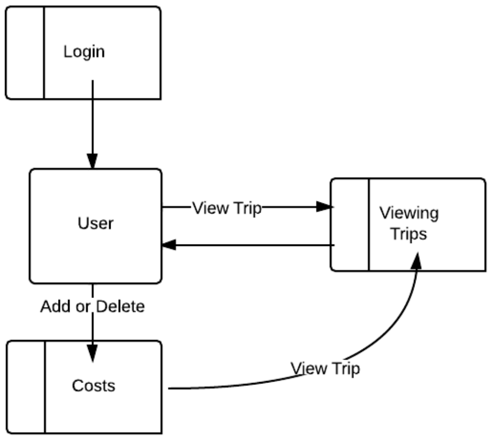
5.3. Users and session


Is signed in?
Yes
Skip the sign up page. Redirect user to his/her trip
No
Show sign in page
Can sign in?
No
error message for invalid emil/password
Yes
Create a session Redirerect user to his/her trip
Option
Allow the user to sign up for an account
Invalid values
sign up
was able to sign up
Create a session Redirerect user to his/her trip

5.4. <u>State Machine</u>



5.5. <u>Context Diagram</u>



6. Functionality and features

[Primary author: Ido Efrati]

6.1. User’s features:

Sign up – users can create an account on Tripper.
Sign in – users can sign in to Tripper if they have an account.
Sign out – users can clear their session. After a sign out a user will be require a new sign in.

6.2. Trip’s features:

Create trip - user can create a new road trip as long as the user specifies the required fields.
View trip - user can see more details about a specific road trip.
Edit trip - user can update road trips’ name.
Cancel a trip- user can delete a road trip from the system.

6.3. Cost’s features:

Add cost- users can submit their monetary and non-monetary contributions to the road trip.
Delete cost- users can remove their monetary and non-monetary contributions from a road trip
View cost - users can see more details about a specific cost.

6.4. Shared features:

Invite friends – users can to share a road trip with other users.
Exchange rate - enables trippers to specify a translation of non-monetary contribution to Tokens, to allow for a more general definition of “cost”.

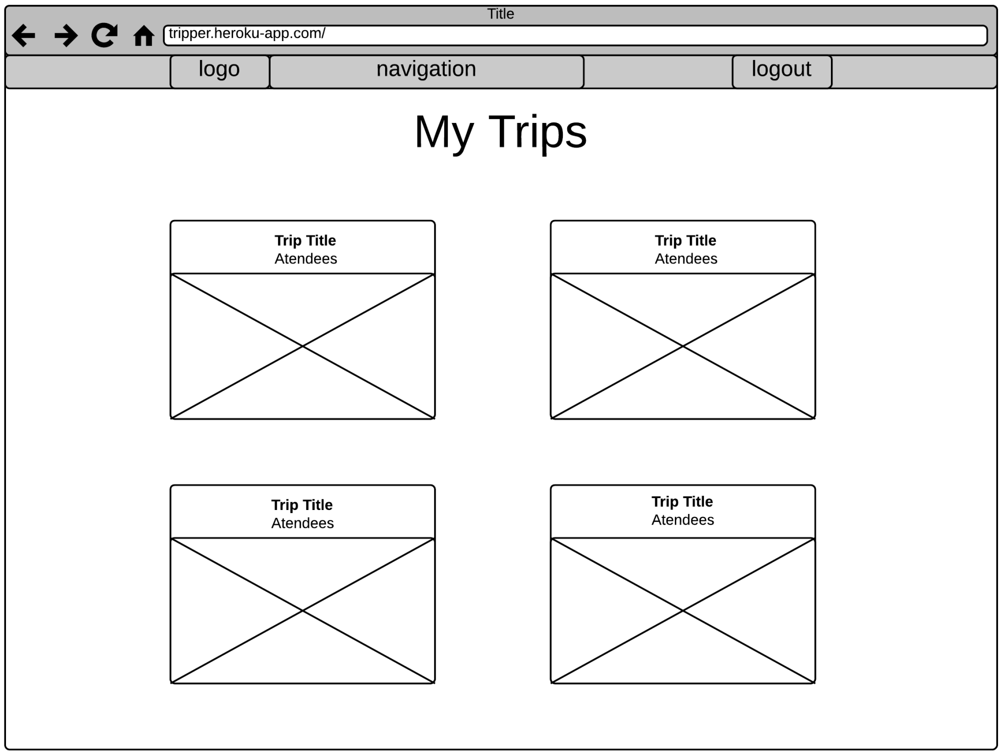
6.5. Splitting features:

Splitting cost - splits the road trips costs evenly among trippers. Users will know how many tokens they own to other users or how many tokens other users owe them. Tokens will be translated to monetary at the end of the road trip.
Analytics - users can view the total trip cost distribution and their own personal cost distribution.

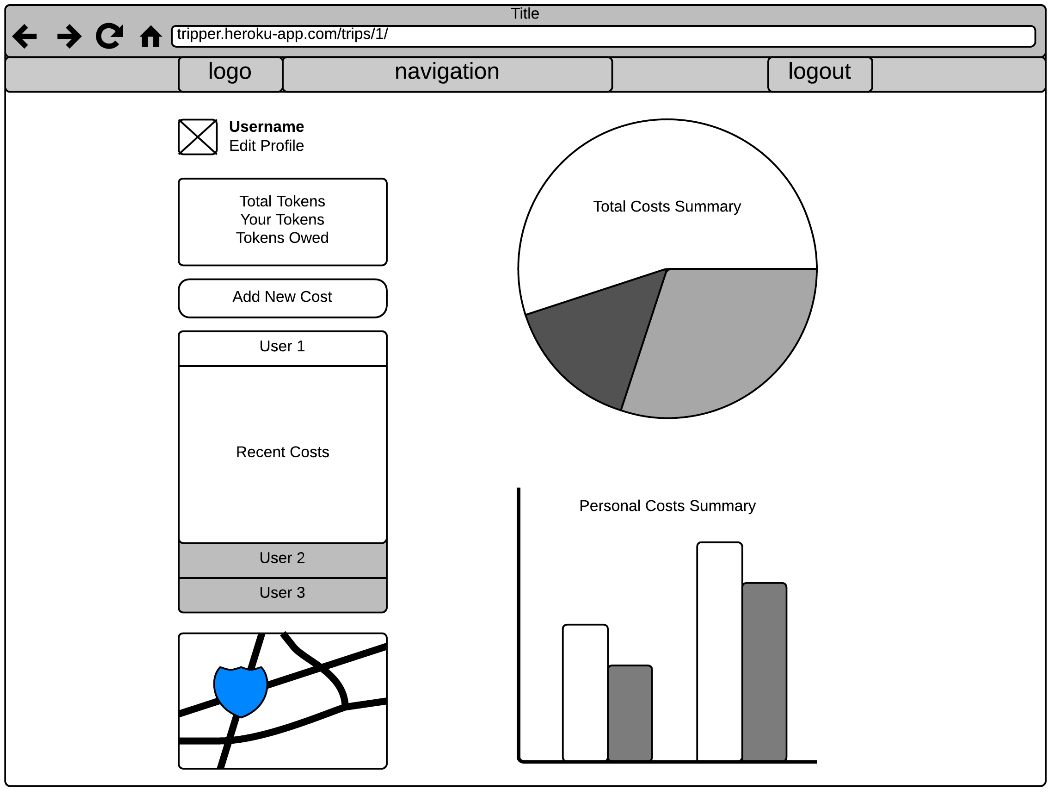
7. User Interface and Wireframe

[Primary author: Ryan Lacey]

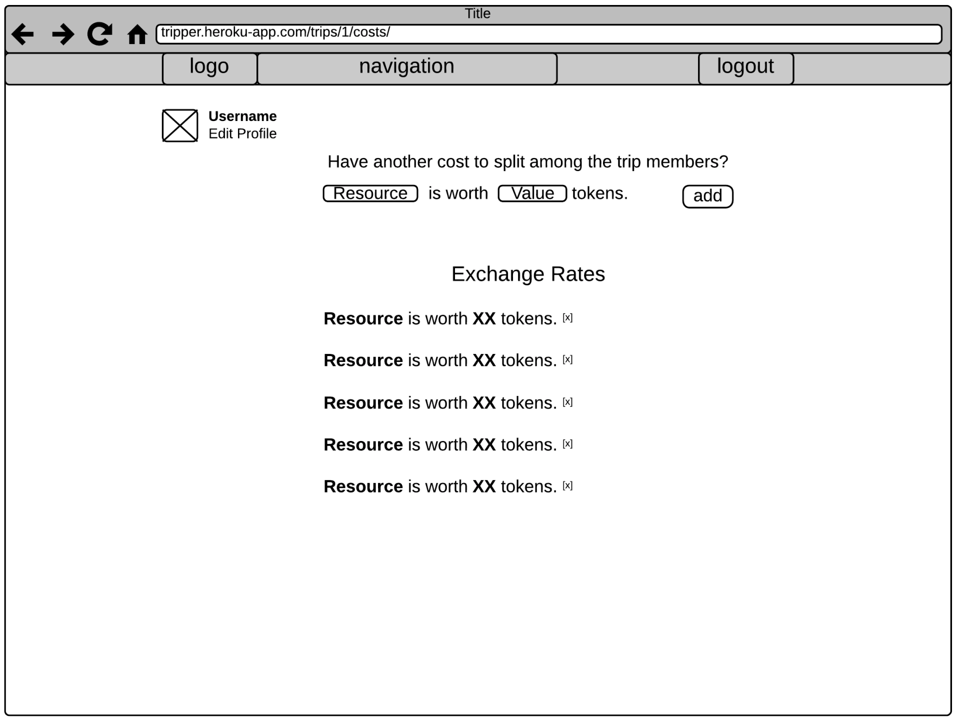
7.1. All trips page:



7.2. Specific trip page:



7.3. Exchange rate page:



Title

tripper.heroku-app.com/trips/1/costs/

Username
Edit Profile

Have another cost to split among the trip members?

Resource is worth Value tokens. add

Exchange Rates

Resource is worth **XX** tokens. [x]

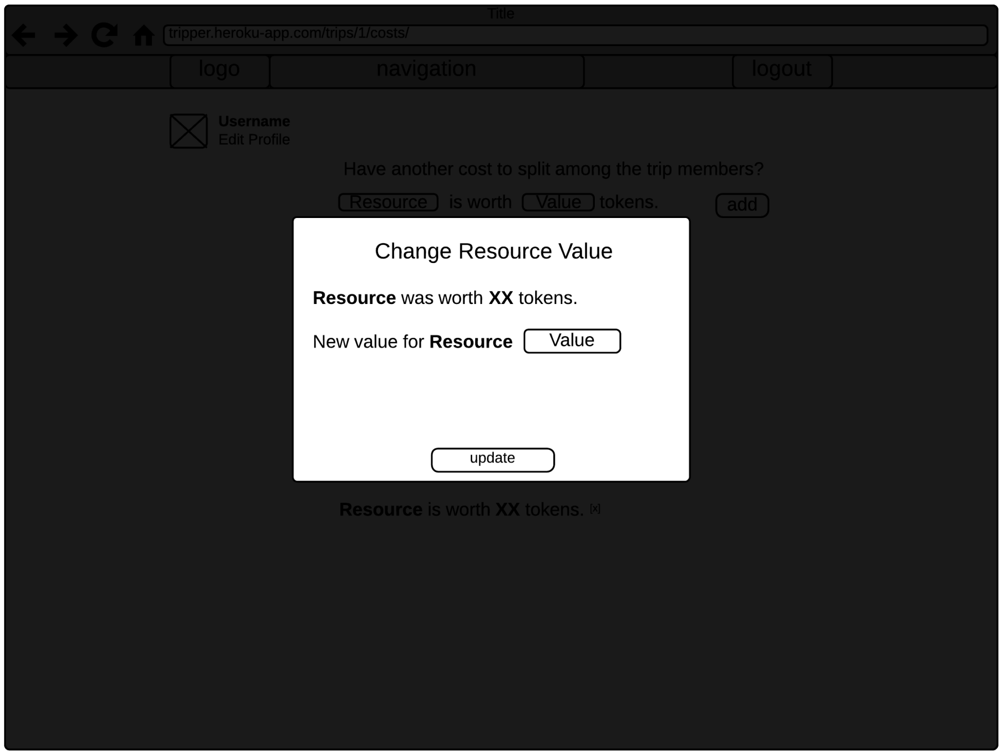
Resource is worth **XX** tokens. [x]

Resource is worth **XX** tokens. [x]

Resource is worth **XX** tokens. [x]

Resource is worth **XX** tokens. [x]

7.4. Update form modal:



Change Resource Value

Resource was worth **XX** tokens.

New value for **Resource** Value

8. Security

[Primary author: Ryan Lacey]

8.1. Authentications

8.1.1. User authentication

When a user tries to log in the server will authenticate their username and password with the username and encrypted password that are stored in the database. Only if the two match will the user be granted access and assigned a session.

8.1.2. Session authentication

When the server verifies if a user is already signed in, it will compare the encrypted session that is stored in the User table under `remember_token` to an encrypted version of the local cookie. Only if the two match will a user be granted access to his or her notes. By doing so the server guarantees that the cookie was not tampered with and prevents unauthorized access.

8.2. Access control:

Users cannot tamper with the URL in an attempt to gain access to a trip that they do not own or that they were not explicitly made a group member of. If a user tries to access a link before a session was established, then he or she will not be able to access the site. If the user is already logged in and tries to change the URL path to access a different trip (i.e. changing the id value in the URL) he or she will be redirected to their respective trips listing*.*

8.3. Functionality safety:

Users can view costs submitted by other members in the same trip, but can only remove costs that they themselves have submitted and can only submit costs under their own name. Members of a trip can edit trip details, but only the trip owner has permission to delete the trip. A trip's members are fixed when a trip is created, so there is no risk of anyone viewing information you have submitted that you had not originally intended to share with. Access security based off table data. Protection from standard web attacks (SQL injection, cross site scripting, cross site request forgery, etc.) is provided by Rails framework.