

Go-Split

Split costs, not friendships

• • •

Harry, Jenna, Ana, Scott and Naeem

Agenda:

1. Project Overview
2. Ways of Working
3. UML Diagrams
4. Entity Relationship
5. Wireframe
6. Demo - Functionalities
7. Challenges & Bugs
8. Future Recommendations
9. Q&A

Project Overview

The Task: Design and build a full-stack application providing a service/solve a problem. Full Stack incorporates both front and back ends.

The API: A report generation for trips outlining the costs due by the organiser. Linking endpoints to the application to add, edit and delete data

Our Project: A mobile application allowing users to add friends, add trips and split costs between friends who attend the trip.



Ways of Working

Trello - Planning tasks

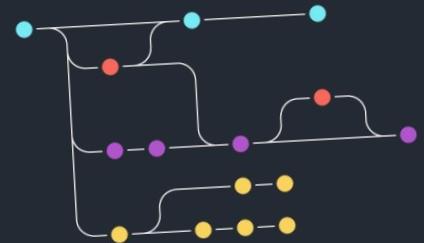
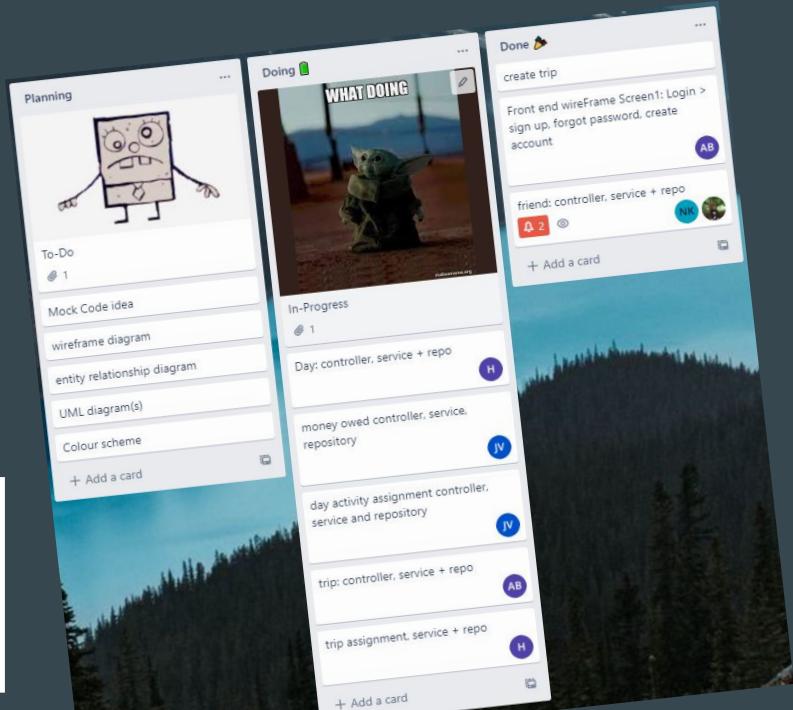
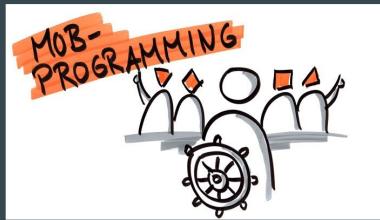
Lucid - Relationship & UML Diagrams

Figma - Wireframing

VS Code

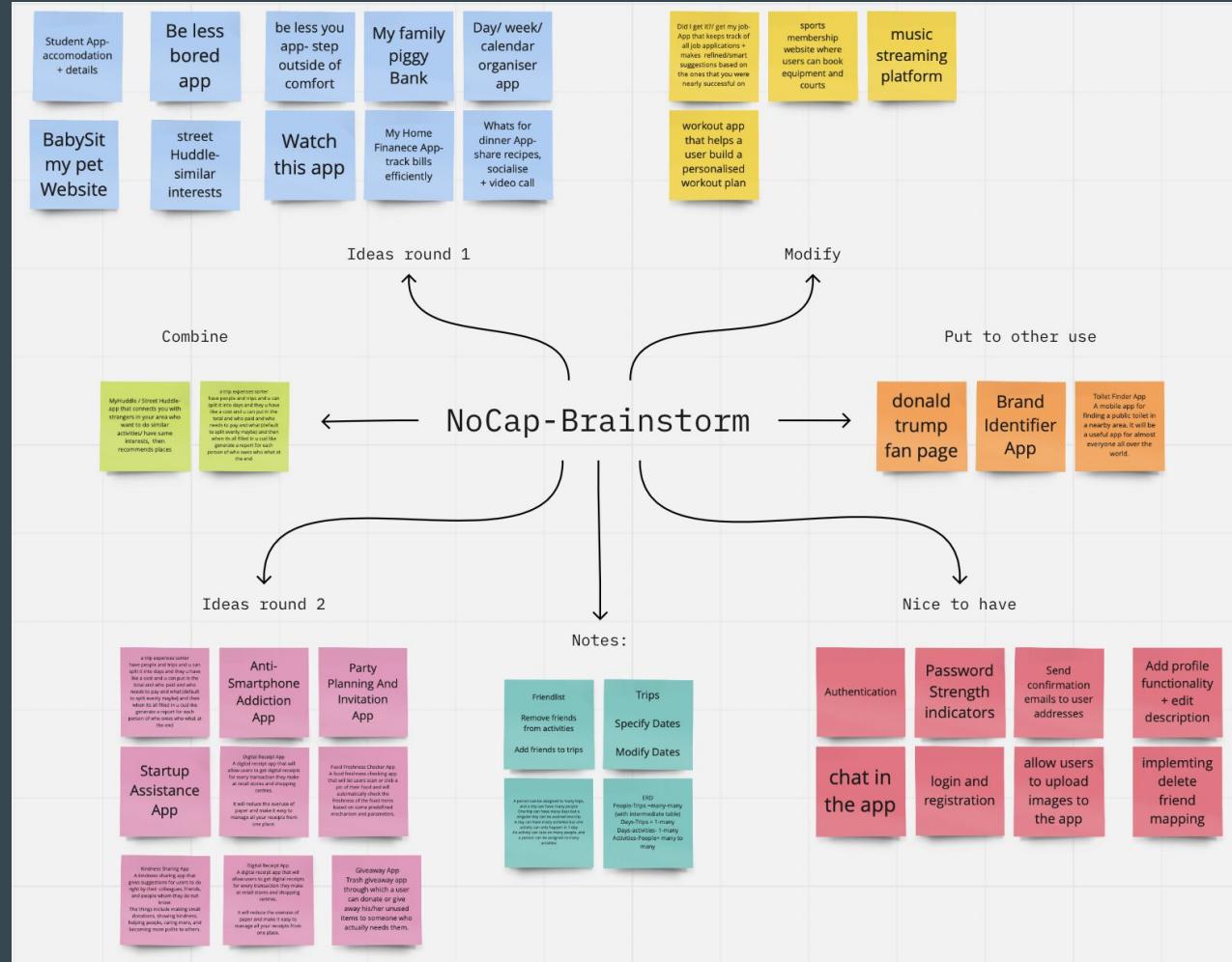
IntelliJ, Driver/Navigator

Git-Branching in VS Code & IntelliJ

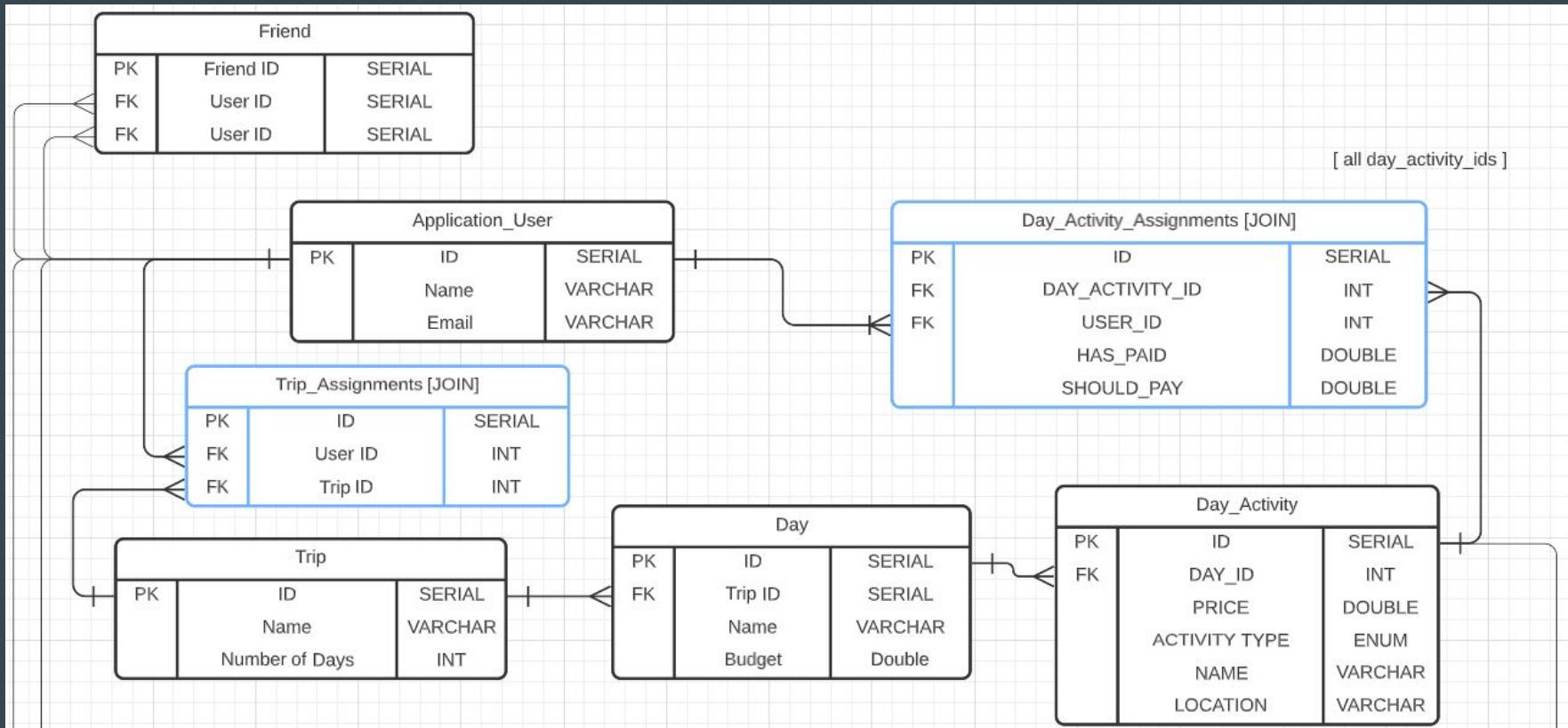


Brainstorm

- Generate
- Combine
- Modify

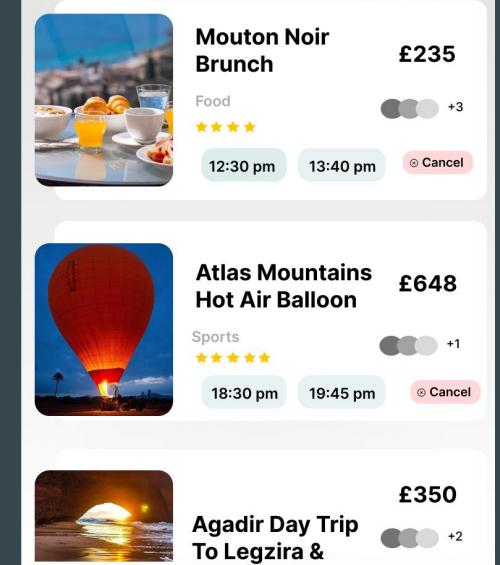
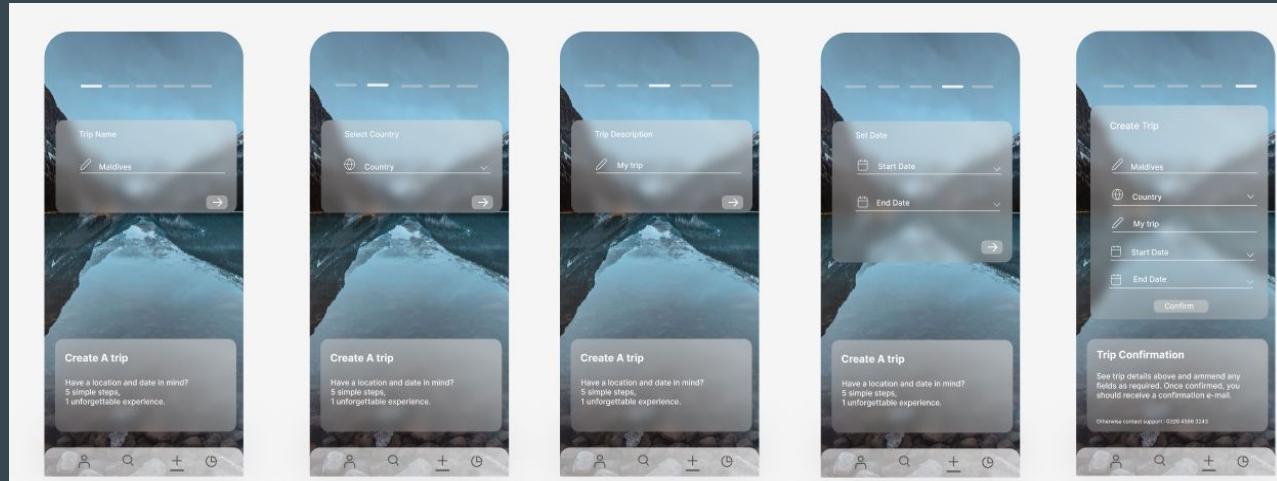
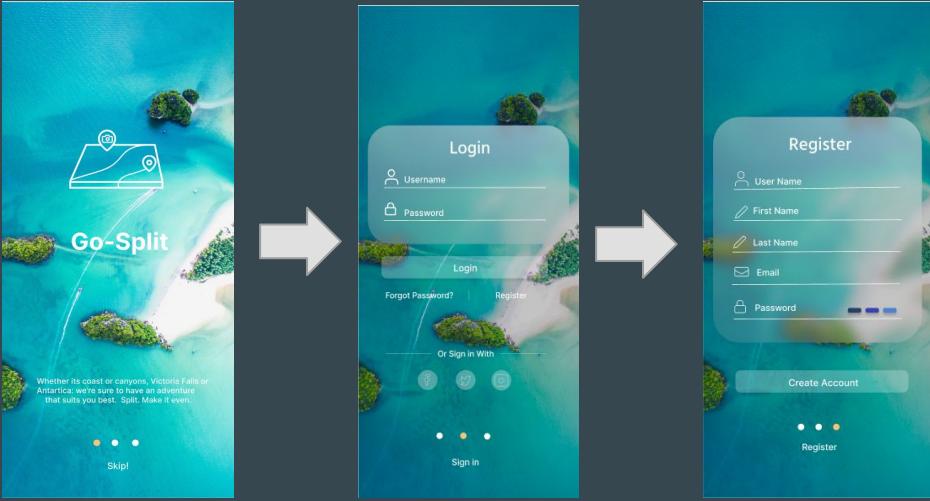


Entity Relationship Diagram- Scott



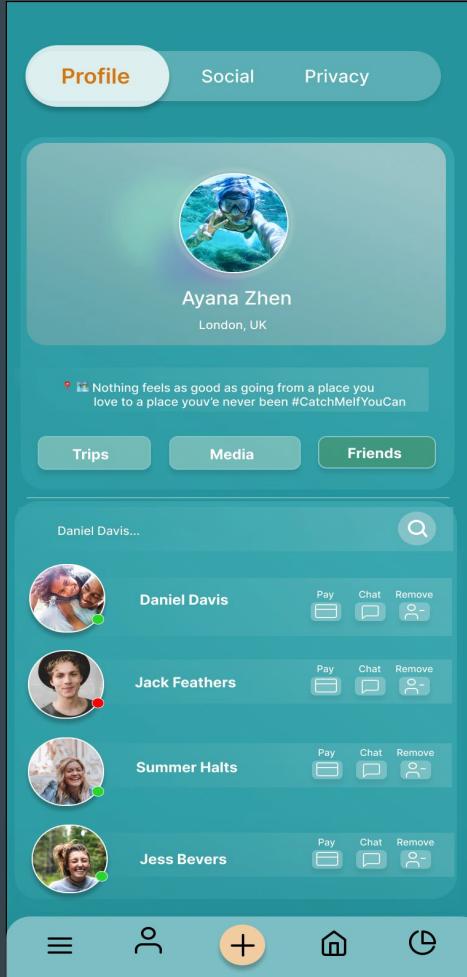
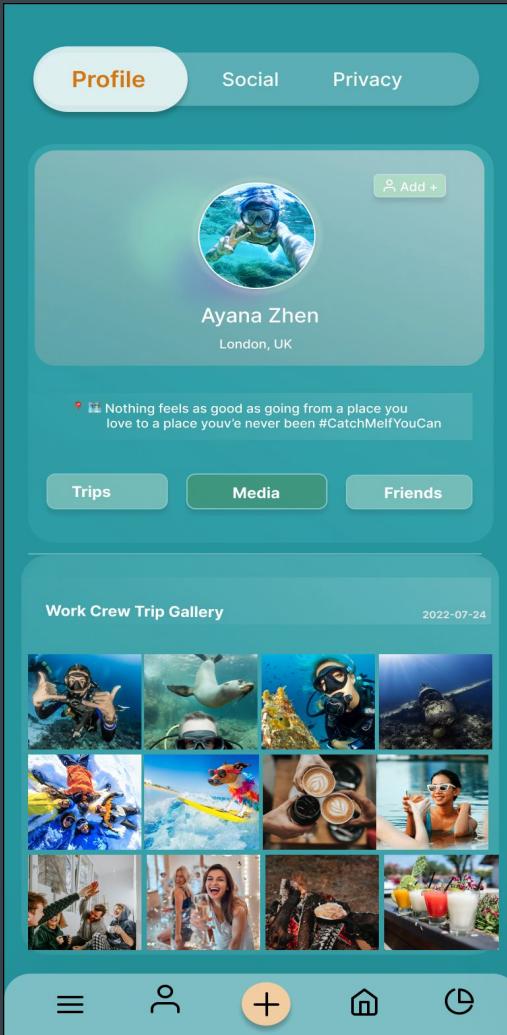
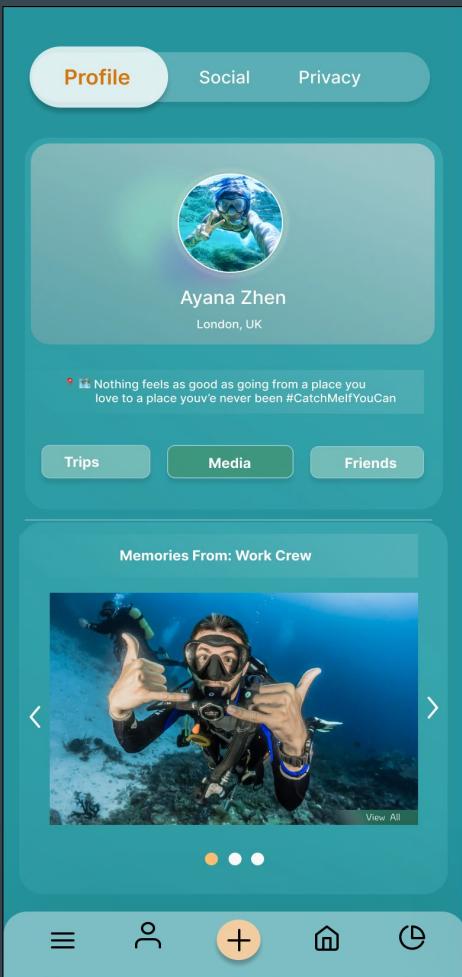
Wireframes

- Sign in
- Register
- Add trip
- View Trip
- View activities



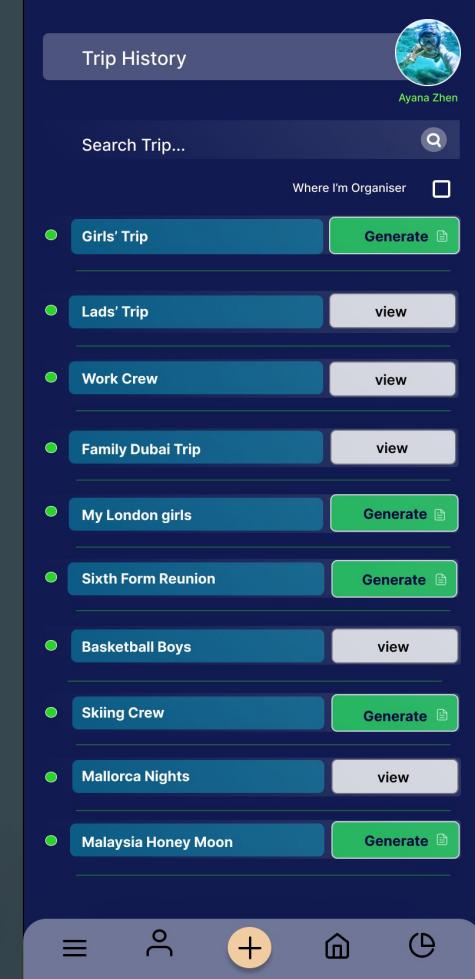
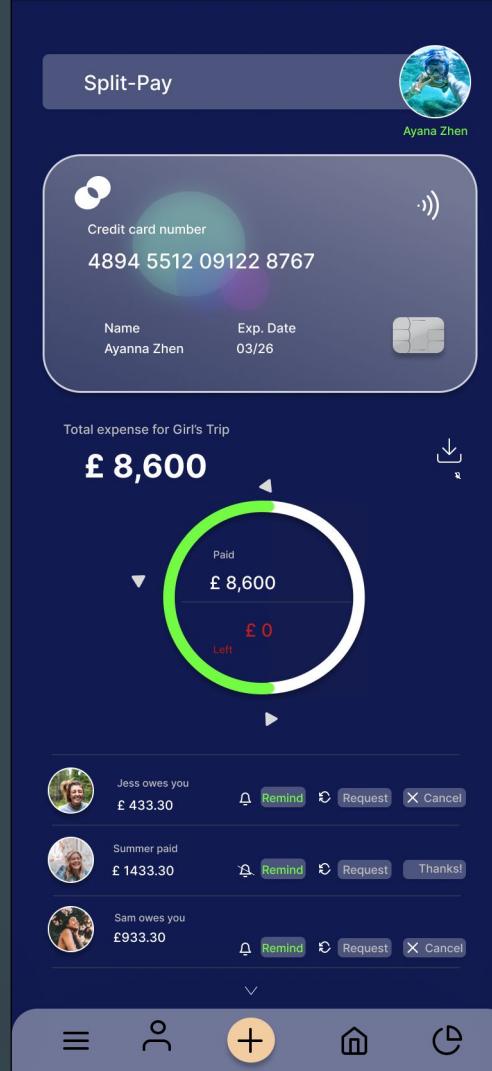
Profile Screens

- View Trips
- Upload Media
- View Gallery
- View friends

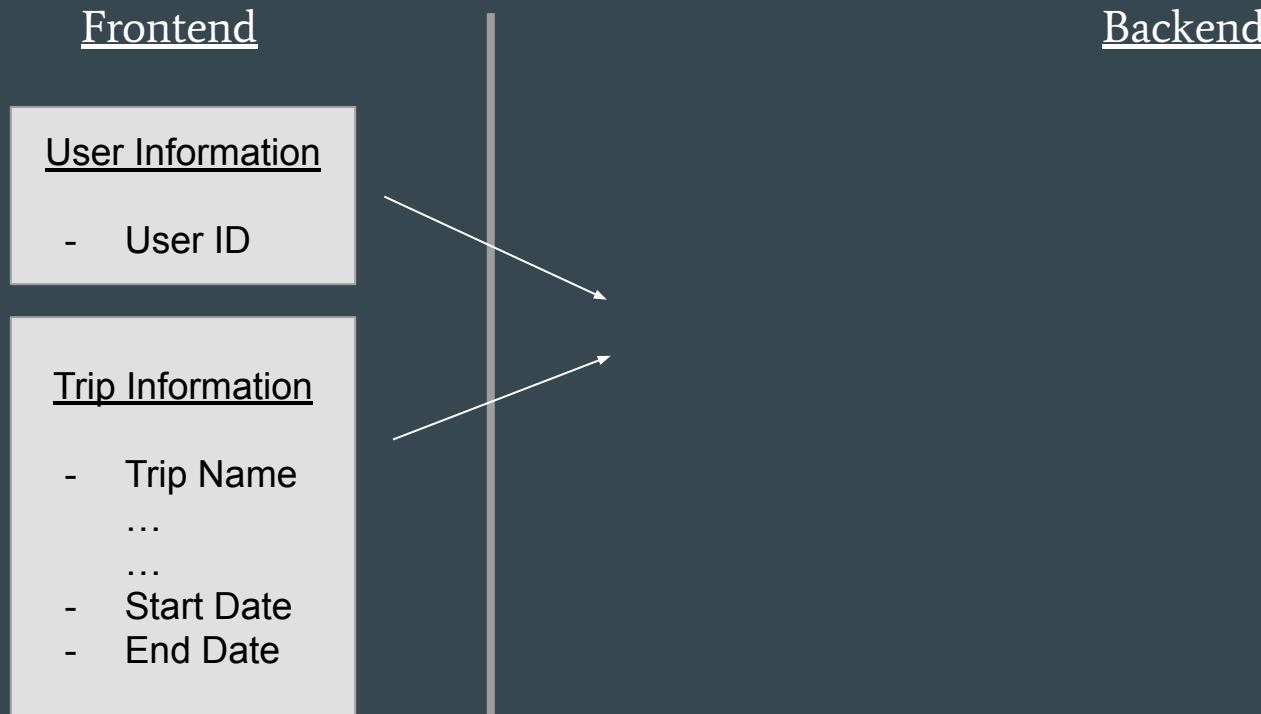


Finance Screens

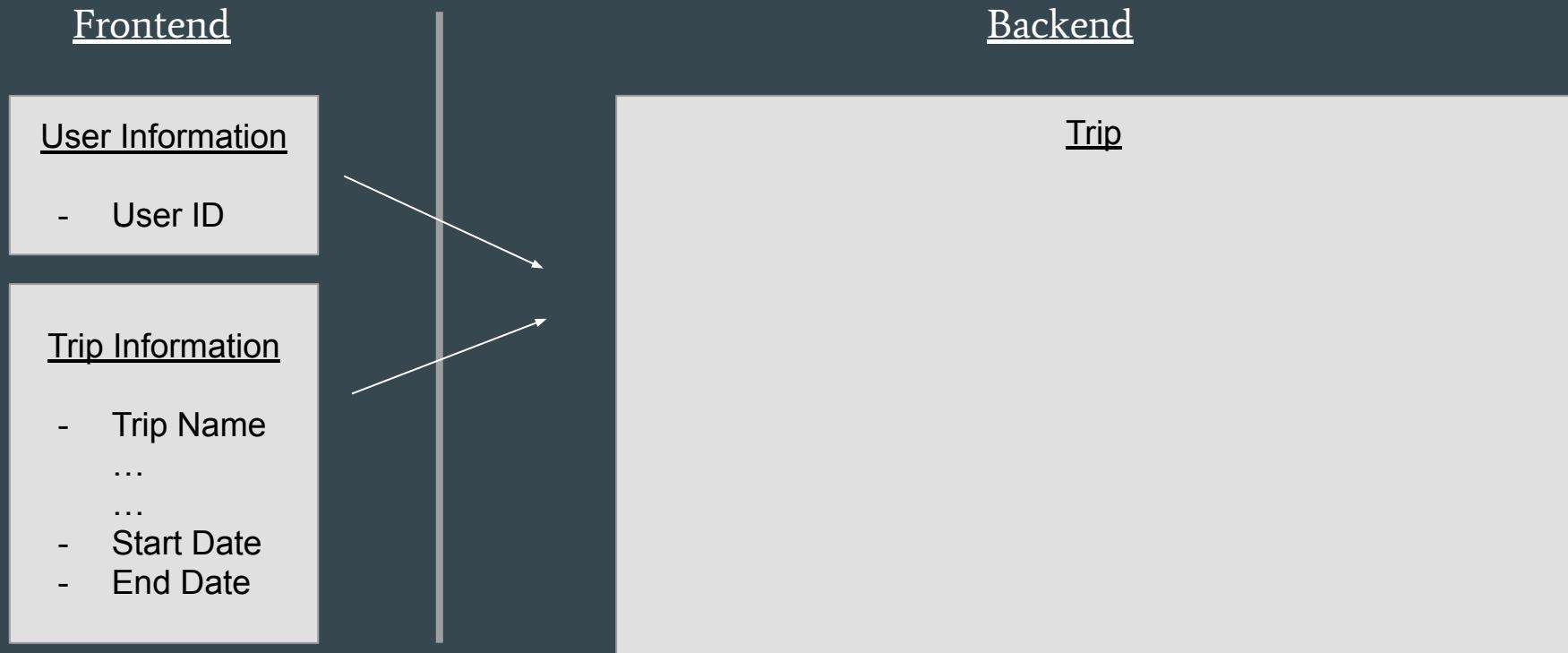
- Generate Pay Reports
- Request Payment
- Remind Friends
- Improve your finance



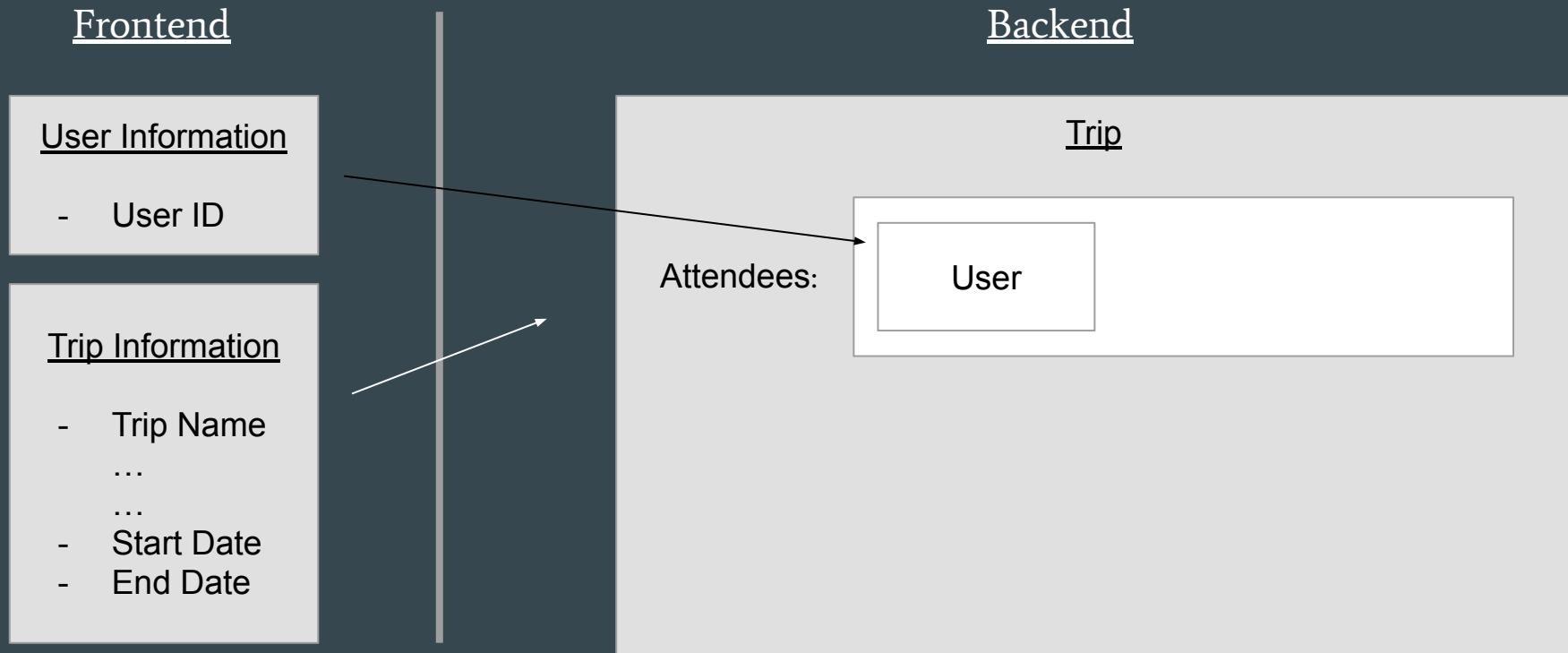
Creating a Trip



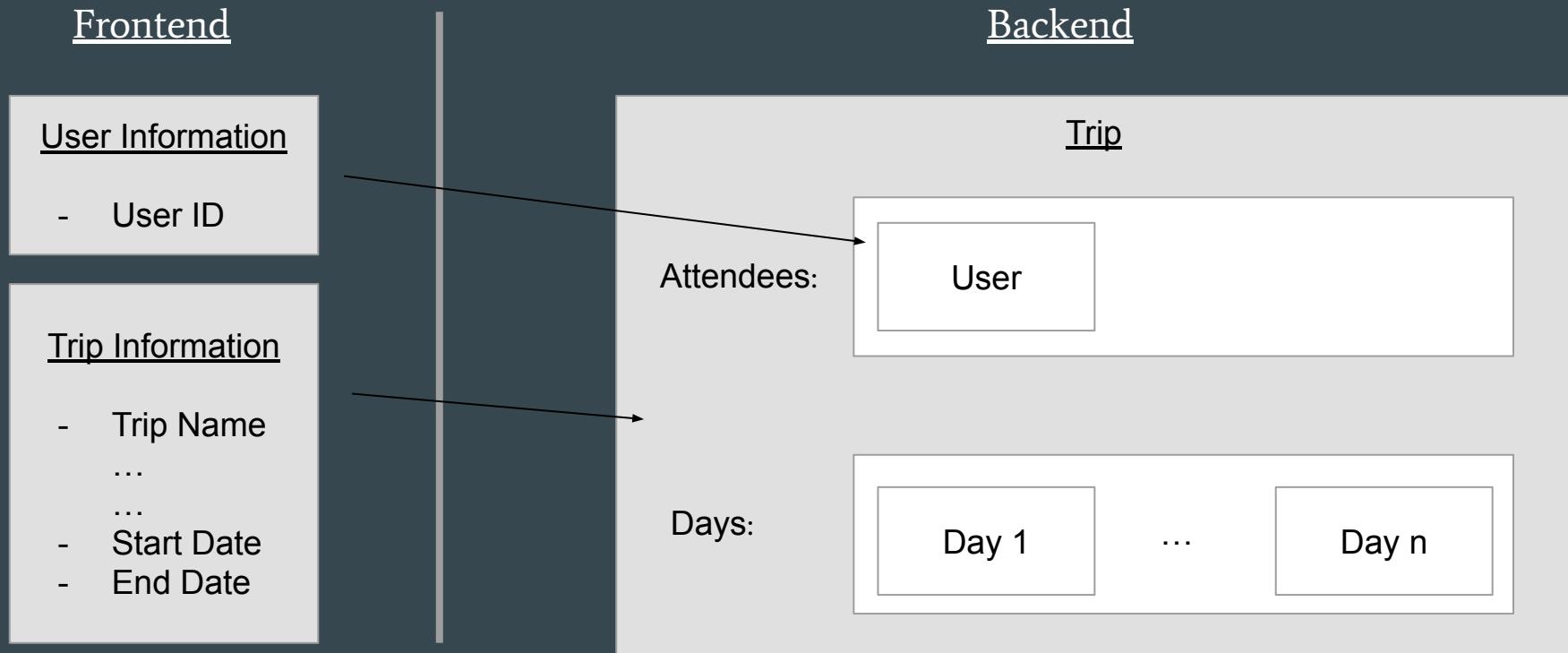
Creating a Trip



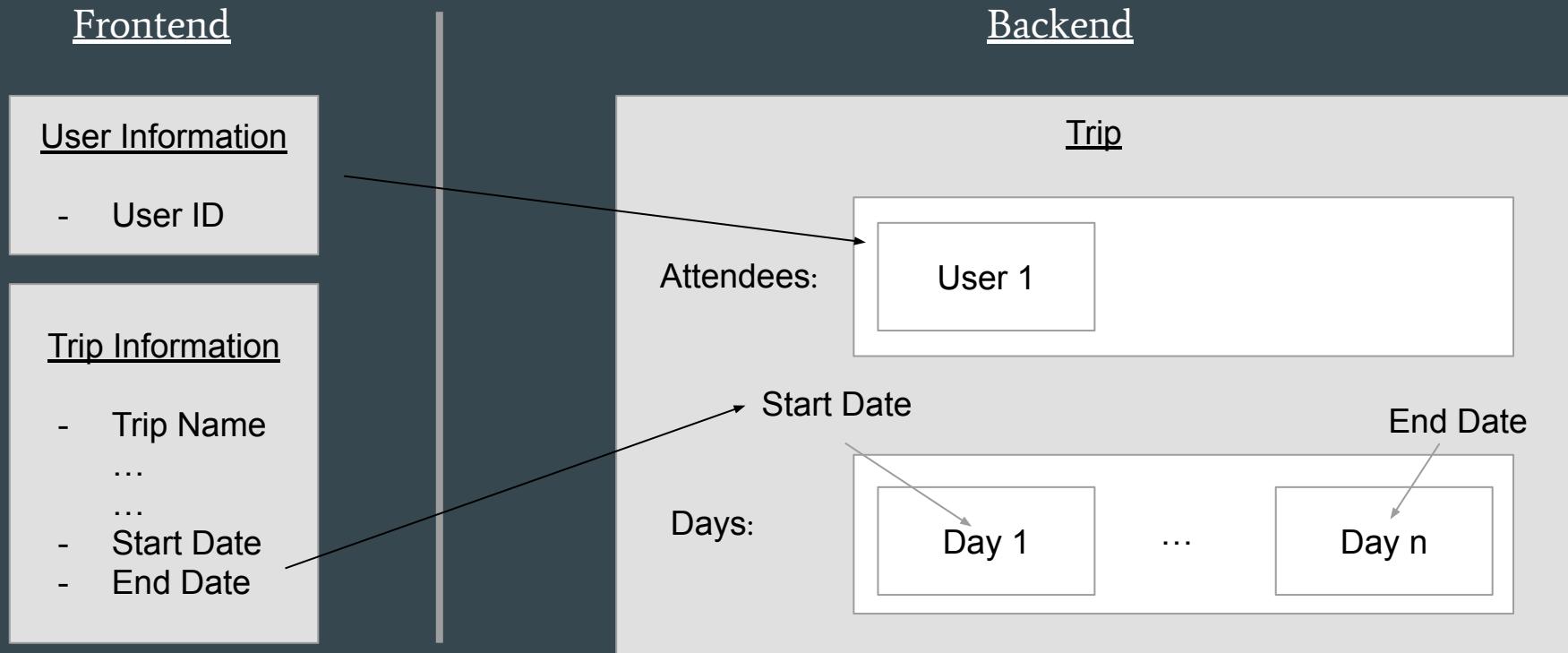
Creating a Trip



Creating a Trip



Creating a Trip



Trip

Attendees:



Start Date

End Date

Days:



Trip

Attendees:



Days:

End Date

Start Date



Day 1

Activities:

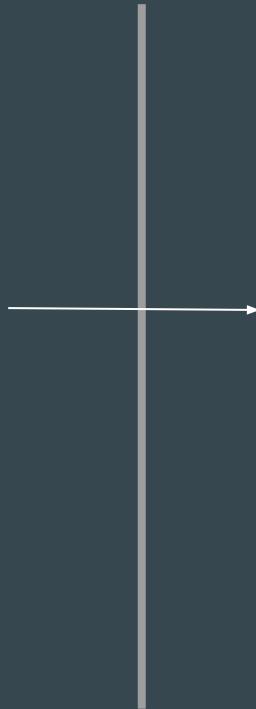
Empty

Backend

Deleting a Trip

Frontend

ID of a
Trip



Deleting a Trip

Frontend

ID of a
Trip

Backend

All User Assignments to this Trip

Trip

Day 1

Day 2

Day 3

...

Day n

Activity 1

Activity 2

...

Activity n

All User Assignments to All Activities

Deleting a Trip

Frontend

ID of a
Trip

Backend

~~All User Assignments to this Trip~~

Trip

Day 1

Day 2

Day 3

...

Day n

Activity 1

Activity 2

...

Activity n

~~All User Assignments to All Activities~~

Deleting a Trip

Frontend

ID of a
Trip

Backend

All User Assignments to this Trip

Trip

Day 1

Day 2

Day 3

...

Day n

Activity 1

Activity 2

...

Activity n

All User Assignments to All Activities

Deleting a Trip

Backend

Frontend

ID of a
Trip

~~All User Assignments to this Trip~~

Trip

Day 1

Day 2

Day 3

Day n

Activity 1

Activity 2

...

Activity n

~~All User Assignments to All Activities~~

Day Code- Ana

Day Class



```
public Day(String name,  
          Double budget,  
          LocalDateTime date,  
          Trip trip) {  
    this.name = name;  
    this.budget = budget;  
    this.date = date;  
    this.trip = trip;  
    this.dayActivities = Sets.newHashSet();  
}
```

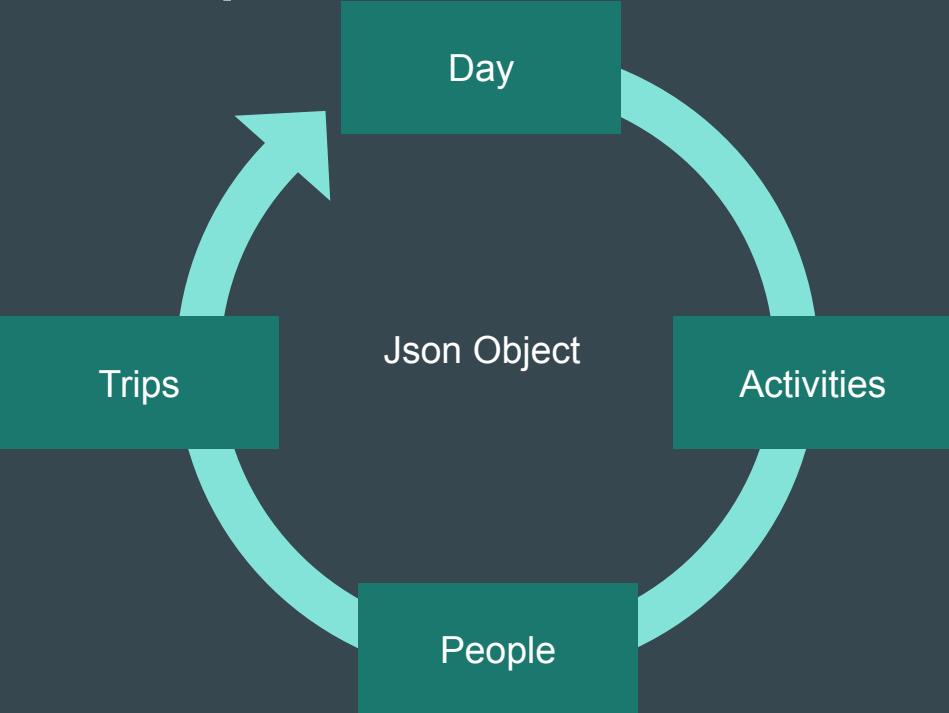
Properties: Trip
and day activities

Day DTO

```
private Long id;  
private String name;  
private Double budget;  
private LocalDateTime date;  
  
public CustomDayDto(Long id,  
                     String name,  
                     Double budget,  
                     LocalDateTime date) {  
    this.id = id;  
    this.name = name;  
    this.budget = budget;  
    this.date = date;  
}
```

No trip or
Activities
properties

Day code - Infinite Loop



Json Object: Infinite Loop

id	first_name	last_name	email	age	group_id	group_name
1	Liam	Smith	lsmith1@sample.com	22	group1	Group 1
2	Olivia	Anderson	oanderson2@sample.com	21	group1	Group 1
3	Noah	Wilson	nwilson3@sample.com	23	group1	Group 1
4	Ava	King	a king4@sample.com	20	group1	Group 1
5	Isabella	Jones	i jones5@sample.com	19	group1	Group 1
6	Lucas	Miller	l miller6@sample.com	24	group1	Group 1
7	Mia	Reed	m reed7@sample.com	21	group1	Group 1
8	Benjamin	Evans	b evans8@sample.com	20	group1	Group 1
9	Charlotte	Clark	c clark9@sample.com	22	group1	Group 1
10	Levi	Howard	l howard10@sample.com	23	group1	Group 1
11	Evelyn	Wong	e wong11@sample.com	21	group1	Group 1
12	Wyatt	Allen	w allen12@sample.com	20	group1	Group 1
13	Sophia	James	s james13@sample.com	22	group1	Group 1
14	Logan	Anderson	l anderson14@sample.com	21	group1	Group 1
15	Harper	Wilson	h wilson15@sample.com	23	group1	Group 1
16	August	King	a king16@sample.com	20	group1	Group 1
17	Emilia	Jones	e jones17@sample.com	19	group1	Group 1
18	Declan	Miller	d miller18@sample.com	24	group1	Group 1
19	Penelope	Reed	p reed19@sample.com	21	group1	Group 1
20	Quinn	Evans	q evans20@sample.com	20	group1	Group 1
21	Madison	Clark	m clark21@sample.com	22	group1	Group 1
22	Greyson	Howard	g howard22@sample.com	23	group1	Group 1
23	Charlotte	Allen	c allen23@sample.com	21	group1	Group 1
24	Scarlett	James	s james24@sample.com	22	group1	Group 1
25	Lincoln	Anderson	l anderson25@sample.com	21	group1	Group 1
26	Eliza	Wilson	e wilson26@sample.com	23	group1	Group 1
27	Wyatt	King	w king27@sample.com	20	group1	Group 1
28	Harper	Jones	h jones28@sample.com	19	group1	Group 1
29	Declan	Miller	d miller29@sample.com	24	group1	Group 1
30	Penelope	Reed	p reed30@sample.com	21	group1	Group 1
31	Quinn	Evans	q evans31@sample.com	20	group1	Group 1
32	Madison	Clark	m clark32@sample.com	22	group1	Group 1
33	Greyson	Howard	g howard33@sample.com	23	group1	Group 1
34	Charlotte	Allen	c allen34@sample.com	21	group1	Group 1
35	Scarlett	James	s james35@sample.com	22	group1	Group 1
36	Lincoln	Anderson	l anderson36@sample.com	21	group1	Group 1
37	Eliza	Wilson	e wilson37@sample.com	23	group1	Group 1
38	Wyatt	King	w king38@sample.com	20	group1	Group 1
39	Harper	Jones	h jones39@sample.com	19	group1	Group 1
40	Declan	Miller	d miller40@sample.com	24	group1	Group 1
41	Penelope	Reed	p reed41@sample.com	21	group1	Group 1
42	Quinn	Evans	q evans42@sample.com	20	group1	Group 1
43	Madison	Clark	m clark43@sample.com	22	group1	Group 1
44	Greyson	Howard	g howard44@sample.com	23	group1	Group 1
45	Charlotte	Allen	c allen45@sample.com	21	group1	Group 1
46	Scarlett	James	s james46@sample.com	22	group1	Group 1
47	Lincoln	Anderson	l anderson47@sample.com	21	group1	Group 1
48	Eliza	Wilson	e wilson48@sample.com	23	group1	Group 1
49	Wyatt	King	w king49@sample.com	20	group1	Group 1
50	Harper	Jones	h jones50@sample.com	19	group1	Group 1
51	Declan	Miller	d miller51@sample.com	24	group1	Group 1
52	Penelope	Reed	p reed52@sample.com	21	group1	Group 1
53	Quinn	Evans	q evans53@sample.com	20	group1	Group 1
54	Madison	Clark	m clark54@sample.com	22	group1	Group 1
55	Greyson	Howard	g howard55@sample.com	23	group1	Group 1
56	Charlotte	Allen	c allen56@sample.com	21	group1	Group 1
57	Scarlett	James	s james57@sample.com	22	group1	Group 1
58	Lincoln	Anderson	l anderson58@sample.com	21	group1	Group 1
59	Eliza	Wilson	e wilson59@sample.com	23	group1	Group 1
60	Wyatt	King	w king60@sample.com	20	group1	Group 1
61	Harper	Jones	h jones61@sample.com	19	group1	Group 1
62	Declan	Miller	d miller62@sample.com	24	group1	Group 1
63	Penelope	Reed	p reed63@sample.com	21	group1	Group 1
64	Quinn	Evans	q evans64@sample.com	20	group1	Group 1
65	Madison	Clark	m clark65@sample.com	22	group1	Group 1
66	Greyson	Howard	g howard66@sample.com	23	group1	Group 1
67	Charlotte	Allen	c allen67@sample.com	21	group1	Group 1
68	Scarlett	James	s james68@sample.com	22	group1	Group 1
69	Lincoln	Anderson	l anderson69@sample.com	21	group1	Group 1
70	Eliza	Wilson	e wilson70@sample.com	23	group1	Group 1
71	Wyatt	King	w king71@sample.com	20	group1	Group 1
72	Harper	Jones	h jones72@sample.com	19	group1	Group 1
73	Declan	Miller	d miller73@sample.com	24	group1	Group 1
74	Penelope	Reed	p reed74@sample.com	21	group1	Group 1
75	Quinn	Evans	q evans75@sample.com	20	group1	Group 1
76	Madison	Clark	m clark76@sample.com	22	group1	Group 1
77	Greyson	Howard	g howard77@sample.com	23	group1	Group 1
78	Charlotte	Allen	c allen78@sample.com	21	group1	Group 1
79	Scarlett	James	s james79@sample.com	22	group1	Group 1
80	Lincoln	Anderson	l anderson80@sample.com	21	group1	Group 1
81	Eliza	Wilson	e wilson81@sample.com	23	group1	Group 1
82	Wyatt	King	w king82@sample.com	20	group1	Group 1
83	Harper	Jones	h jones83@sample.com	19	group1	Group 1
84	Declan	Miller	d miller84@sample.com	24	group1	Group 1
85	Penelope	Reed	p reed85@sample.com	21	group1	Group 1
86	Quinn	Evans	q evans86@sample.com	20	group1	Group 1
87	Madison	Clark	m clark87@sample.com	22	group1	Group 1
88	Greyson	Howard	g howard88@sample.com	23	group1	Group 1
89	Charlotte	Allen	c allen89@sample.com	21	group1	Group 1
90	Scarlett	James	s james90@sample.com	22	group1	Group 1
91	Lincoln	Anderson	l anderson91@sample.com	21	group1	Group 1
92	Eliza	Wilson	e wilson92@sample.com	23	group1	Group 1
93	Wyatt	King	w king93@sample.com	20	group1	Group 1
94	Harper	Jones	h jones94@sample.com	19	group1	Group 1
95	Declan	Miller	d miller95@sample.com	24	group1	Group 1
96	Penelope	Reed	p reed96@sample.com	21	group1	Group 1
97	Quinn	Evans	q evans97@sample.com	20	group1	Group 1
98	Madison	Clark	m clark98@sample.com	22	group1	Group 1
99	Greyson	Howard	g howard99@sample.com	23	group1	Group 1
100	Charlotte	Allen	c allen100@sample.com	21	group1	Group 1
101	Scarlett	James	s james101@sample.com	22	group1	Group 1
102	Lincoln	Anderson	l anderson102@sample.com	21	group1	Group 1
103	Eliza	Wilson	e wilson103@sample.com	23	group1	Group 1
104	Wyatt	King	w king104@sample.com	20	group1	Group 1
105	Harper	Jones	h jones105@sample.com	19	group1	Group 1
106	Declan	Miller	d miller106@sample.com	24	group1	Group 1
107	Penelope	Reed	p reed107@sample.com	21	group1	Group 1
108	Quinn	Evans	q evans108@sample.com	20	group1	Group 1
109	Madison	Clark	m clark109@sample.com	22	group1	Group 1
110	Greyson	Howard	g howard110@sample.com	23	group1	Group 1
111	Charlotte	Allen	c allen111@sample.com	21	group1	Group 1
112	Scarlett	James	s james112@sample.com	22	group1	Group 1
113	Lincoln	Anderson	l anderson113@sample.com	21	group1	Group 1
114	Eliza	Wilson	e wilson114@sample.com	23	group1	Group 1
115	Wyatt	King	w king115@sample.com	20	group1	Group 1
116	Harper	Jones	h jones116@sample.com	19	group1	Group 1
117	Declan	Miller	d miller117@sample.com	24	group1	Group 1
118	Penelope	Reed	p reed118@sample.com	21	group1	Group 1
119	Quinn	Evans	q evans119@sample.com	20	group1	Group 1
120	Madison	Clark	m clark120@sample.com	22	group1	Group 1
121	Greyson	Howard	g howard121@sample.com	23	group1	Group 1
122	Charlotte	Allen	c allen122@sample.com	21	group1	Group 1
123	Scarlett	James	s james123@sample.com	22	group1	Group 1
124	Lincoln	Anderson	l anderson124@sample.com	21	group1	Group 1
125	Eliza	Wilson	e wilson125@sample.com	23	group1	Group 1
126	Wyatt	King	w king126@sample.com	20	group1	Group 1
127	Harper	Jones	h jones127@sample.com	19	group1	Group 1
128	Declan	Miller	d miller128@sample.com	24	group1	Group 1
129	Penelope	Reed	p reed129@sample.com	21	group1	Group 1
130	Quinn	Evans	q evans130@sample.com	20	group1	Group 1
131	Madison	Clark	m clark131@sample.com	22	group1	Group 1
132	Greyson	Howard	g howard132@sample.com	23	group1	Group 1
133	Charlotte	Allen	c allen133@sample.com	21	group1	Group 1
134	Scarlett	James	s james134@sample.com	22	group1	Group 1
135	Lincoln	Anderson	l anderson135@sample.com	21	group1	Group 1
136	Eliza	Wilson	e wilson136@sample.com	23	group1	Group 1
137	Wyatt	King	w king137@sample.com	20	group1	Group 1
138	Harper	Jones	h jones138@sample.com	19	group1	Group 1
139	Declan	Miller	d miller139@sample.com	24	group1	Group 1
140	Penelope	Reed	p reed140@sample.com	21	group1	Group 1
141	Quinn	Evans	q evans141@sample.com	20	group1	Group 1
142	Madison	Clark	m clark142@sample.com	22	group1	Group 1
143	Greyson	Howard	g howard143@sample.com	23	group1	Group 1
144	Charlotte	Allen	c allen144@sample.com	21	group1	Group 1
145	Scarlett	James	s james145@sample.com	22	group1	Group 1
146	Lincoln	Anderson	l anderson146@sample.com	21	group1	Group 1
147	Eliza	Wilson	e wilson147@sample.com	23	group1	Group 1
148	Wyatt	King	w king148@sample.com	20	group1	Group 1
149	Harper	Jones	h jones149@sample.com	19	group1	Group 1
150	Declan	Miller	d miller150@sample.com	24	group1	Group 1
151	Penelope	Reed	p reed151@sample.com	21	group1	Group 1
152	Quinn	Evans	q evans152@sample.com	20	group1	Group 1
153	Madison	Clark	m clark153@sample.com	22	group1	Group 1
154	Greyson	Howard	g howard154@sample.com	23	group1	Group 1
155	Charlotte	Allen	c allen155@sample.com	21	group1	Group 1
156	Scarlett	James	s james156@sample.com	22	group1	Group 1
157	Lincoln	Anderson	l anderson157@sample.com	21	group1	Group 1
158	Eliza	Wilson	e wilson158@sample.com	23	group1	Group 1
159	Wyatt	King	w king159@sample.com	20	group1	Group 1
160	Harper	Jones	h jones160@sample.com	19	group1	Group 1
161	Declan	Miller	d miller161@sample.com	24	group1	Group 1
162	Penelope	Reed	p reed162@sample.com	21	group1	Group 1
163	Quinn	Evans	q evans163@sample.com	20	group1	Group 1
164	Madison	Clark	m clark164@sample.com	22	group1	Group 1
165	Greyson	Howard	g howard165@sample.com	23	group1	Group 1
166	Charlotte	Allen	c allen166@sample.com	21	group1	Group 1
167	Scarlett	James	s james167@sample.com	22	group1	Group 1
168	Lincoln	Anderson	l anderson168@sample.com	21	group1	Group 1
169	Eliza	Wilson	e wilson169@sample.com	23	group1	Group 1
170	Wyatt	King	w king170@sample.com	20	group1	Group 1
171	Harper	Jones	h jones171@sample.com	19	group1	Group 1
172	Declan	Miller	d miller172@sample.com	24	group1	Group 1
173	Penelope	Reed	p reed173@sample.com	21	group1	Group 1
174	Quinn	Evans	q evans174@sample.com	20	group1	Group 1
175	Madison	Clark	m clark175@sample.com	22	group1	Group 1
176	Greyson	Howard	g howard176@sample.com	23	group1	Group 1
177	Charlotte	Allen	c allen177@sample.com	21	group1	Group 1
178	Scarlett	James	s james178@sample.com	22	group1	Group 1
179	Lincoln	Anderson	l anderson179@sample.com	21	group1	Group 1
180	Eliza	Wilson	e wilson180@sample.com	23	group1	Group 1
181	Wyatt	King	w king181@sample.com	20	group1	Group 1
182	Harper	Jones	h jones182@sample.com	19	group1	Group 1
183	Declan	Miller	d miller183@sample.com	24	group1	Group 1
184	Penelope	Reed	p reed184@sample.com	21	group1	Group 1
185	Quinn	Evans	q evans185@sample.com	20	group1	Group 1
186	Madison	Clark	m clark186@sample.com	22	group1	Group 1
187	Greyson	Howard	g howard187@sample.com	23	group1	Group 1
188	Charlotte	Allen	c allen188@sample.com	21	group1	Group 1
189	Scarlett	James	s james189@sample.com	22	group1	Group 1
190	Lincoln	Anderson	l anderson190@sample.com	21	group1	Group 1
191	Eliza	Wilson	e wilson191@sample.com	23	group1	Group 1
192	Wyatt	King	w king192@sample.com	20	group1	Group 1
193	Harper	Jones	h jones193@sample.com	19	group1	Group 1
194	Declan	Miller	d miller194@sample.com	24	group1	Group 1
195	Penelope	Reed	p reed195@sample.com	21	group1	Group 1
196	Quinn	Evans	q evans196@sample.com	20	group1	Group 1
197	Madison	Clark	m clark197@sample.com	22	group1	Group 1
198	Greyson	Howard	g howard198@sample.com	23	group1	Group 1
199	Charlotte	Allen	c allen199@sample.com	21	group1	Group 1
200	Scarlett	James	s james200@sample.com	22	group1	Group 1
201	Lincoln	Anderson	l anderson201@sample.com	21	group1	Group 1
202	Eliza	Wilson	e wilson202@sample.com	23	group1	Group 1
203	Wyatt	King	w king203@sample.com	20	group1	Group 1
204	Harper	Jones	h jones204@sample.com	19	group1	Group 1
205						

Json Object

```
    {
      "id": 4,
      "name": "abhishek",
      "user_id": 6,
      "created_at": "2015-06-04 23:33:59",
      "updated_at": "2015-06-02 23:33:59",
      "users": {
        "id": 6,
        "name": "abhishek",
        "last_name": "deshkar",
        "email": "abhi.alone@gmail.com",
        "cell_no": "9979559210",
        "gender": "male",
        "country": "91",
        "primary_nick": "abhishek",
        "created_at": "2015-06-04 23:33:59",
        "updated_at": "2015-06-15 00:02:40"
      }
    }
  }
```

Day Activity Code- Jenna

```
public List<PayeeAndPayer> generateActivityCostByUser(Long userID, Long dayActivityID) {  
    List<DayActivityAssignment> dayActivityAssignments = dayActivityAssignmentRepository  
        .returnActivityAssignmentsByActivityID(dayActivityID);
```

```
@GetMapping("/generateOwingFromTrip")  
public List<PayeeAndPayer> generateTripCosts(@RequestParam Long tripID) {
```

- Goes through an activity and finds out who owes what to each person by activity
- Goes through all activities of a trip and adds up what each person owes each other person
- Then cancels out any owing that goes both ways!

Friends Code - Scott FrontEnd

```
const addFriend = async (fA) => {  
  
  const currentUserUsername = "thelifeandtimesofnaeem"  
  try {  
    const friends = await axios  
      .post('http://localhost:8080/friend/addFriend/${currentUserUsername}/${fA.username}')  
    const data = friends.data;  
    setFriends(data)  
    document.getElementById(`#${fA.id}addFriendBtn`).innerHTML = "Added";  
  } catch (err) {  
    setFriends([])  
    document.getElementById(`#${fA.id}friendBTNclick`).innerHTML = `${err.response.data.message}`;  
  }  
}
```

```
const searchUsers = async (e) => {  
  const username = e.target.value;  
  try {  
    const users = await axios.get(`http://l  
  const data = users.data;  
  setUsers(data)
```

```
<div id="users-list">  
{  
  users.map((user, index) => {  
    return (  
      <div key={index} id="one-user">  
        <div id="user-p-container">  
          <div className="search-user-image-container">  
            <img src={user.imgURL} alt="" className="user-image-style"/>  
          </div>  
          <h1 id="headingUser">{user.firstname} {user.lastname}</h1>  
          <p id="pUsername">{user.username}</p>  
          <button id={`${user.id}addFriendBtn`} className="addFriendBtn" onClick={() => addFriend(user)}>Add Friend</button>  
        </div>  
        <div className="err-return">  
          <p id={`${user.id}friendBTNclick`} className="friendBtnclick"></p>  
        </div>  
      </div>  
    )  
  })  
}
```

In the user list div, each item “one-user” is mapped by the data stored in users with a unique key (index) corresponding to each user

Add friend

- Made asynchronous so addFriend will only load once the searchUser method declared earlier loads the list of users
- addFriend called onClick, linked to each userId, which will prompt either an error message to load below the profile if unsuccessful or the button will be altered to say “added” if successful

Friends Code - BackEnd

```
@PostMapping("/addFriend/{currentUserUsername}/{friendToAddUsername}")
public String addFriend(@PathVariable("currentUserUsername") String currentUserUsername,
                       @PathVariable("friendToAddUsername") String friendToAddUsername
) throws Exception{
    return friendService.addFriend(currentUserUsername, friendToAddUsername);
}
```

```
public String addFriend(String currentUserUsername, String friendToAddUsername) {
    ApplicationUser userC = userRepository.getUserByName(currentUserUsername);
    ApplicationUser userF = userRepository.getUserByName(friendToAddUsername);

    Friend friendTA = new Friend(userC, userF, currentUserUsername, friendToAddUsername);

    Friend existingUsernameCombo1 = friendRepository.findFriendPairUsername(currentUserUsername, friendToAddUsername);
    Friend existingUsernameCombo2 = friendRepository.findFriendPairUsername(friendToAddUsername, currentUserUsername);
}
```

```
if (userF == null) {
    throw new InvalidDataAccessApiUsageException("User does not exist! Check the details are right and try again, " + userC.getFirstName() + " " + userC.getLastName());
} else if (userC == userF) {
    throw new InvalidDataAccessApiUsageException("Oi, big boy! You can't be friends with yourself! " + userC.getFirstName() + " " + userC.getLastName());
} else if (existingUsernameCombo1 == null && existingUsernameCombo2 == null) {
    friendRepository.save(friendTA);
} else {
    throw new InvalidDataAccessApiUsageException("You're already friends with " + userF.getFirstName() + "!");
}

return "You have added " + userF.getFirstName() + " " + userF.getLastName() + " to your friend list, " + userC.getFirstName() + " " + userC.getLastName();
}
```



Add Friend:

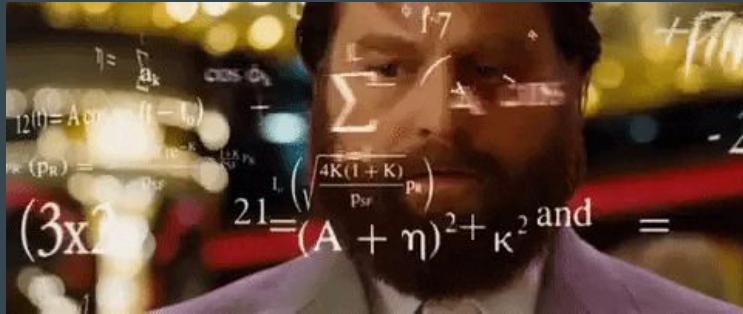
- Define PathVariable to connect service
- Call addFriend method
- User data in order of person using it and person on who they want to add
- Friend repository calls a method to check if friend pair exists.
- Added extra functionality to prevent user from adding themselves
- Prevented duplicate friends being added.

- LIVE DEMO- Jenna

Challenges & Bugs

CSS Styling...

i.e. THE BANE
OF OUR
EXISTENCE



Data Loader



Keeping the project in scope



The infinite JSON loop...



Future Recommendations

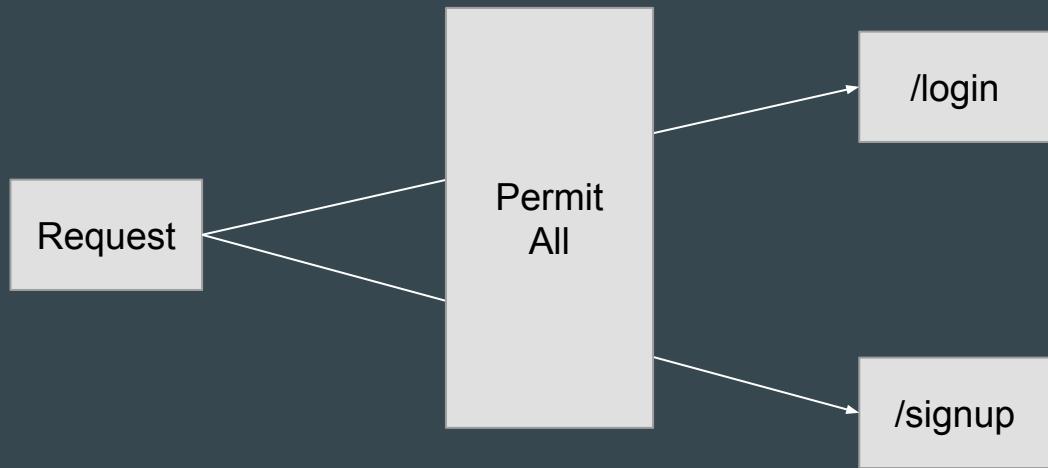
Delete Mappings

- Enable removing friends in your user profile
- Removal of activities from any day

Authentication

- We would have made more of our code dynamic, e.g.:
 - Removing the hard-coded “logged-in” user
 - Integrated the log-in feature with the app so that, upon submitting the log-in form, it would load the corresponding user profile

User Authentication Code



http

```
.authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry  
.antMatchers(HttpMethod.POST, ...antPatterns: "/authenticate") ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl  
.permitAll() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry  
.antMatchers(HttpMethod.POST, ...antPatterns: "/sign_up") ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl  
.permitAll() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry  
.antMatchers(HttpMethod.GET, ...antPatterns: "/test") ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl  
.permitAll();
```

/login

```
try {
    authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(
        username,
        password
    ));
    return authenticationUserDetailsService.loadUserByUsername(username);
}
```

```
/login
```

```
try {
    authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(
        username,
        password
    ));
    return authenticationUserDetailsService.loadUserByUsername(username);
}
```

```
Set<SimpleGrantedAuthority> authorities =
    user.getOwner() ? APP_OWNER.getGrantedAuthorities() :
    user.getAdmin() ? APP_ADMIN.getGrantedAuthorities() :
                      APP_USER.getGrantedAuthorities();

return new AuthenticationUserImplUserDetails(
    user.getUsername(),
    user.getPassword(),
    authorities,
    isAccountNonExpired: true,
    isAccountNonLocked: true,
    isCredentialNonExpired: true,
    isEnabled: true
);
```

```
/login
```

```
try {
    authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(
        username,
        password
    ));
    return authenticationUserDetailsService.loadUserByUsername(username);
}
```

```
Set<SimpleGrantedAuthority> authorities =
    user.getOwner() ? APP_OWNER.getGrantedAuthorities() :
    user.getAdmin() ? APP_ADMIN.getGrantedAuthorities() :
        APP_USER.getGrantedAuthorities();

return new AuthenticationUserImplUserDetails(
    user.getUsername(),
    user.getPassword(),
    authorities,
    isAccountNonExpired: true,
    isAccountNonLocked: true,
    isCredentialNonExpired: true,
    isEnabled: true
);
```

```
APP_ADMIN(Sets.newHashSet(
    APP_ADMIN_READ_SELF,
    APP_ADMIN_WRITE_SELF,
    APP_USER_READ_ALL,
    APP_USER_WRITE_ALL
)),
```

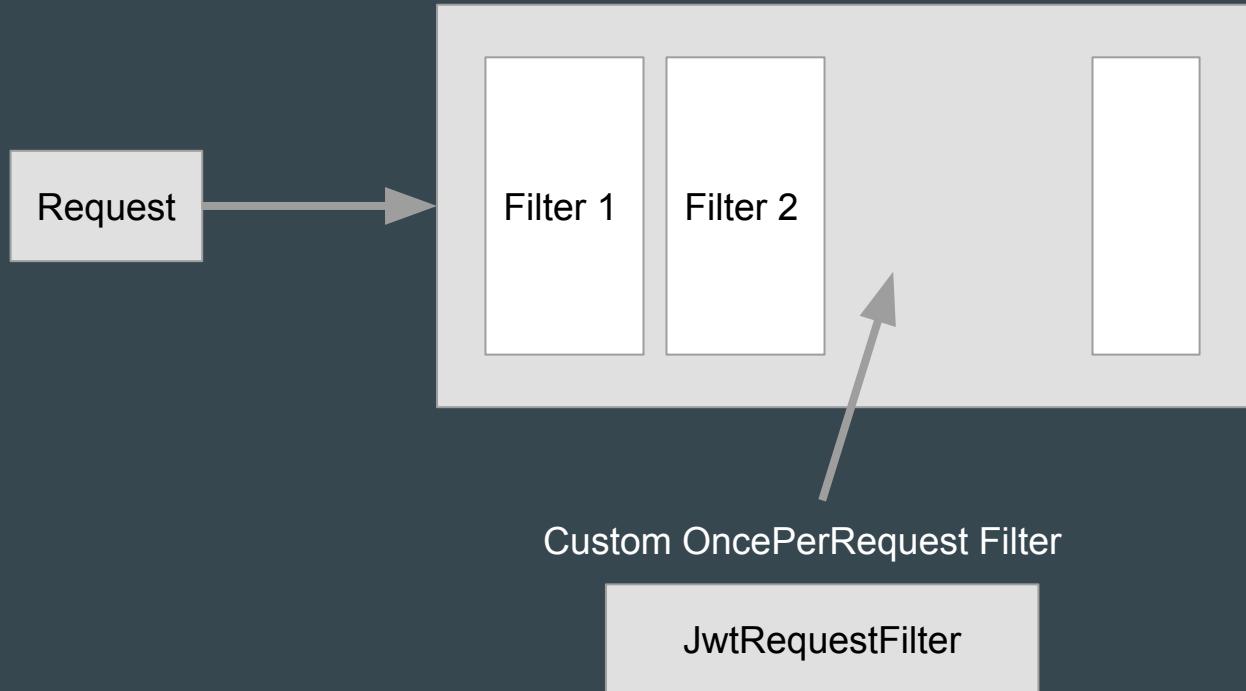
```
return Jwts  
    .builder()  
    .setHeaderParam("typ", "JWT")  
    .setSubject(userDetails.getUsername())  
    .claim("authorities", userDetails.getAuthorities())  
    .setIssuedAt(new Date())  
    .setExpiration(java.sql.Date.valueOf(LocalDate.now().plusDays(jwtConfig.getTokenExpirationAfterDays())))  
    .signWith(Keys.hmacShaKeyFor(jwtConfig.getSecretKey().getBytes()))  
    .compact();
```

```
{  
  "success": true,  
  "message": "Token generated",  
  "payload": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.  
eyJzdWIiOiJhZG1pbjEiLCJhdXRob3JpdGllcyI6W3siYXV0aG9yaXR5Ij  
oiYXBwX3VzZXI6cmVhZF9hbGwifSx7ImF1dGhvcmI0eSI6ImFwcF91c2Vy  
OndyaXR1X2FsbCJ9LHSiYXV0aG9yaXR5IjoiYXBwX2FkbWluOnJ1YWfc2  
VsZiJ9LHSiYXV0aG9yaXR5IjoiYXBwX2FkbWluOndyaXR1X3NlbGYifV0s  
ImIhdCI6MTY1NzIyNjE1MywiZXhwIjoxNjU3MjM4NDAwfQ.  
Sqodc2iFv1FyHKX0k6ErxdXhWZlkfUYnJX0DI7NdTj0q6G_0L6TmcNHWWGM  
iEkH88nfuCdnGaM07s-0xzAmoCcQ"  
}
```

```
{  
  "success": true,  
  "message": "Token generated",  
  "payload": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.  
eyJzdWIiOiJhZG1pbjEiLCJhdXRob3JpdGllcyI6W3siYXV0aG9yaXR5Ij  
oiYXBwX3VzZXI6cmVhZF9hbGwifSx7ImF1dGhvcm10eSI6ImFwcF91c2Vy  
OndyaXR1X2FsbCJ9LHSiYXV0aG9yaXR5IjoiYXBwX2FkbWluOnJlYWRfc2  
VsZiJ9LHSiYXV0aG9yaXR5IjoiYXBwX2FkbWluOndyaXR1X3NlbGYifV0s  
ImlhCI6MTY1NzIyNjE1MywiZXhwIjoxNjU3MjM4NDAwfQ.  
Sqodc2iFv1FyHKX0k6ErdXhWZ1kfUYnJX0DI7NdTj0q6G_0L6TmcNHWWGM  
iEkH88nfuCdnGaM07s-0xzAmoCcQ"
```

```
{  
  "sub": "admin1",  
  "authorities": [  
    {  
      "authority": "app_user:read_all"  
    },  
    {  
      "authority": "app_user:write_all"  
    },  
    {  
      "authority": "app_admin:read_self"  
    },  
    {  
      "authority": "app_admin:write_self"  
    }  
  "iat": 1657226153,  
  "exp": 1657238400  
}
```

Spring Security Filter Chain



```
Jws<Claims> claimsJws = Jwts.parser()
    .setSigningKey(Keys.hmacShaKeyFor(jwtConfig.getSecretKey().getBytes()))
    .parseClaimsJws(token);
```

```
if (expiration.before(new Date())) {
    filterChain.doFilter(request, response);
    return;
}
```

Set of
authorities

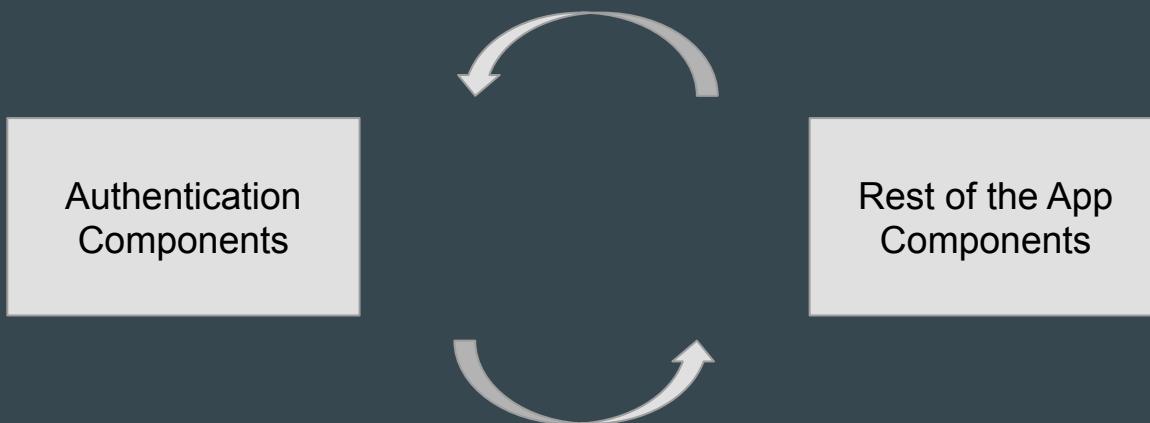
```
SecurityContextHolder.getContext().setAuthentication(authentication);
```

```
} catch (JwtException e) {
    throw new IllegalStateException(String.format("Token %s is invalid.", token));
}
```

```
@GetMapping("/get_all")
@PreAuthorize("hasAuthority('app_user:read_all'))
```

```
import React, { useState } from 'react'
```

```
const [authenticatedUser, setAuthenticatedUser] = useState(null)
```



Thanks for listening