Studious

Ruby Off the Rails

Joel Diener
Alex Tareshawty
Kyle Thompson
Sean Whitehurst
Mary Zhou

What is Studious?

Studious is a web app that allows students to connect in common academic interest groups, share flashcards, and chat!

Technologies Used

- Rails 5
- React
- Redux

Rails 5

- API-Only Mode
- PostgreSQL
- Testing with FactoryGirl

Action Cable

- Used for simple integration of WebSockets into Rails
- Allowed for real-time chat implementation

React

- Javascript UI library
- React vs Plain Javascript
  - One-way data flow
  - Virtual DOM
  - JSX

React Example

```
import React from 'react';

import auth from '../authentication';

import './landing_page.scss';

const { components } = auth;
const { Signup } = components;

const LandingPage = ({ actions }) => (
  <div className="LandingPage">
    <h1 id="LandingPage-header">Find academic groups. Chat. Practice flash cards.</h1>
    <Signup onSignup={(user) => actions.signupUser(user)}/>
  </div>
);

export default LandingPage;
```

React Example

```
class Chat extends Component {

  ...

  render() {
    const { messages } = this.props;

    if (!messages) {
      return (
        <div>Loading...</div>
      );
    }

    return (
      <div className="Chat">
        <div className="Chat-messages" ref="messagesContainer" >
          { messages.map(({ createdAt, id, content, user }) =>
            <Message createdAt={createdAt} id={id} content={content} user={user} />) }
        </div>
        <div className="Chat-input">
          <textarea className="Chat-input--textbox" ref="message" onKeyDown={this.handleKeyDown}/>
          <button className="Chat-input--sendButton" onClick={this.handleClick}>Send</button>
        </div>
      </div>
    );
  }
}
```

Redux

- SPOCOC
- App state in one place
- State connection

Redux Example

```
const mapDispatchToProps = (dispatch) => ({
  createSubscription: bindActionCreators(actions.createSubscription, dispatch),
  loadMessages: bindActionCreators(actions.loadMessages, dispatch),
  receiveMessage: bindActionCreators(actions.receiveMessage, dispatch)
});

const mapStateToProps = (state, { id }) => ({
  consumer: chatChannel.selectors.consumer(state),
  messages: selectors.messagesById(state, id),
  subscription: selectors.subscriptionById(state, id)
});

class Chat extends Component {

  ...

}

export default connect(mapStateToProps, mapDispatchToProps)(Chat);
```

Routes Implementation

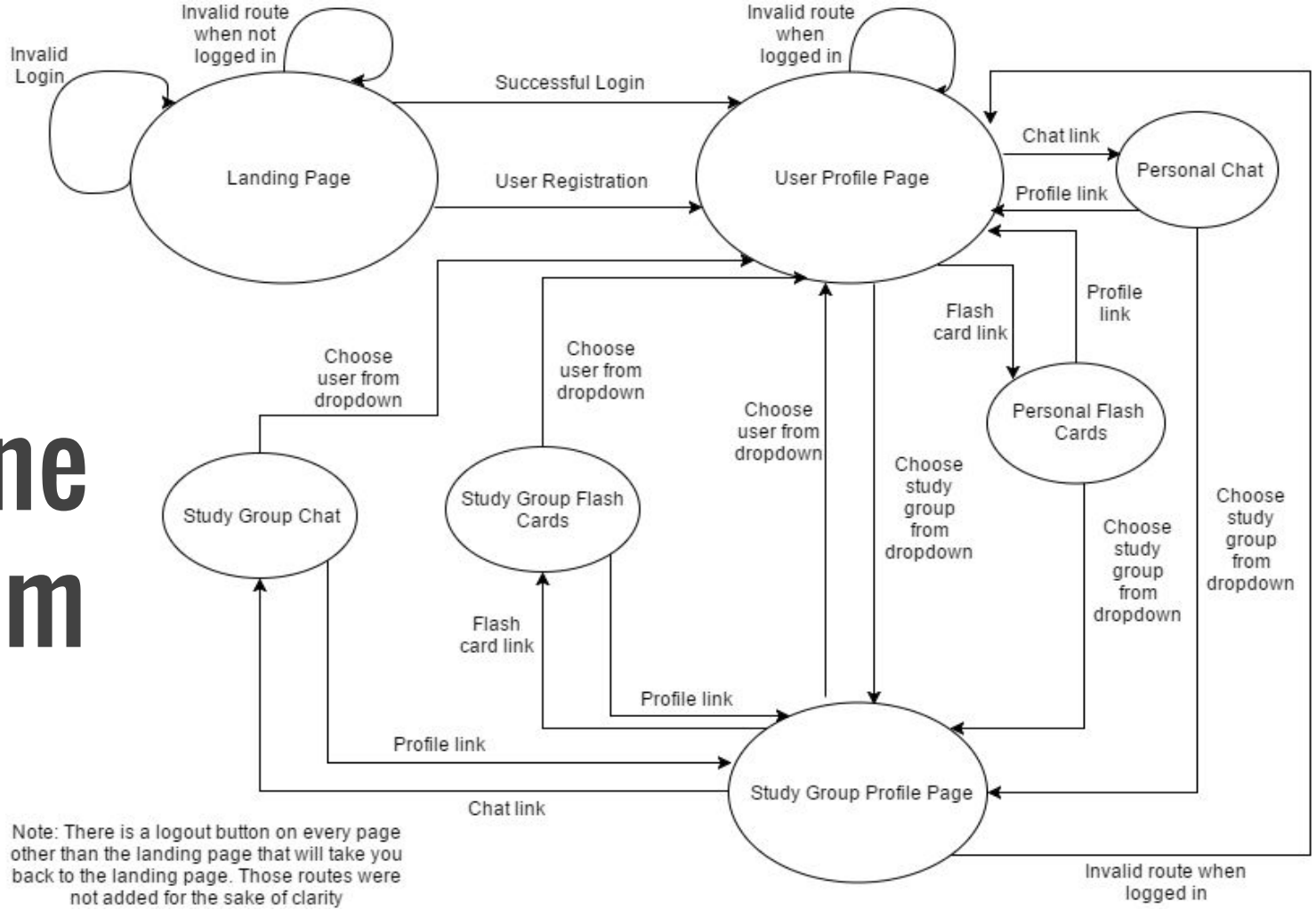
- Resources
- JBuilder

```
mount ActionCable.server => '/sockets/:token'

post 'user_token' => 'user_token#create'
get 'chatrooms/:chatroom_id/messages' => 'messages#index'

resources :users, only: [:index, :create, :update, :destroy]
resources :users, only: [:show] do
  resources :flash_card_sets, only: [:create]
end
```

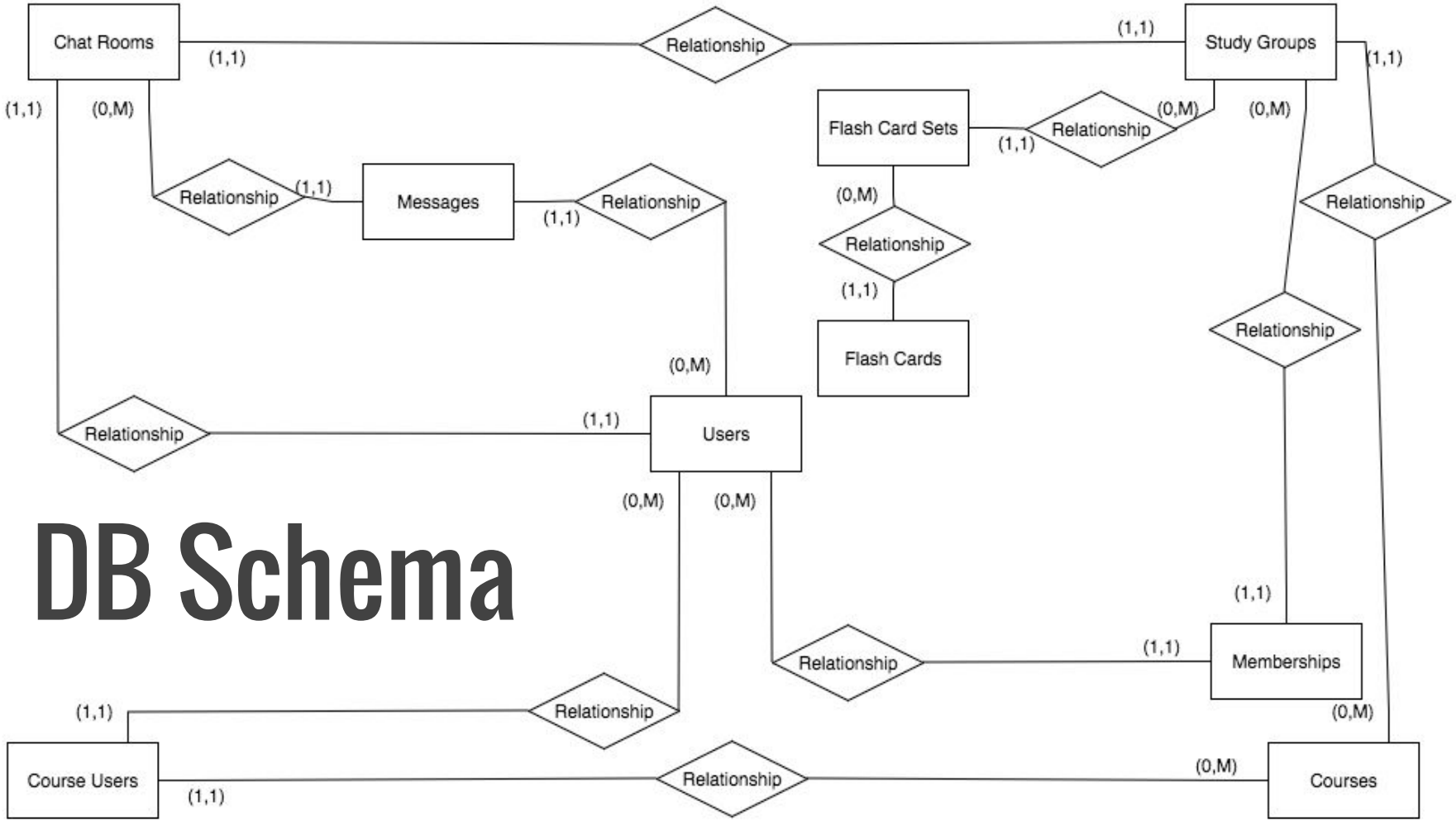
JBuilder View

```
json.id flash_card_set.id
json.name flash_card_set.name
json.studyGroupId flash_card_set.study_group_id
json.userId flash_card_set.user_id

json.flashCards do
  json.array! flash_card_set.flash_cards.each do |flash_card|
    json.partial! 'api/flash_cards/flash_card', flash_card: flash_card
  end
end
```

State Machine Diagram



Note: There is a logout button on every page other than the landing page that will take you back to the landing page. Those routes were not added for the sake of clarity


Chat Rooms
Relationship
Study Groups
(1,1)
(1,1)
(1,1)
(1,1)
(0,M)
(0,M)
Flash Card Sets
Relationship
(0,M)
(1,1)
(0,M)
Relationship
(1,1)
Messages
Relationship
(1,1)
Relationship
(0,M)
Relationship
(1,1)
Flash Cards
Relationship
(0,M)
Relationship
(1,1)
Users
(0,M)
(0,M)
(1,1)
Relationship
(1,1)
Memberships
(0,M)
(1,1)
Relationship
Course Users
(1,1)
Relationship
(0,M)
Courses

DB Schema

Controller Action

- Each controller action renders the appropriate JBuilder view

```ruby
def update
  @flash_card_set ||= FlashCardSet.find params[:id]

  if @flash_card_set.update_attributes update_params
    render :show, status: 200
  else
    render json: { resource: "flashCardSet", errors: @flash_card_set.errors }, status: 409
  end
end
```

Demo

Questions?