THE OHIO STATE UNIVERSITY

“BuckeyeWhoKnows” BBS Website

Project Team 6
Rui Ge, Wei Tang, Ruiyang Liu, Yan Zhang, Keenan Fraly

Content

- Background
- Demo
- Technical Details
- Test
- Furthermore

Background

- Sharing useful information
- Some specific places
- Student and Staff

Background

- Trading Used Books
- Traffic Status
- Events
- Finding Teammates/Roommates
- Group Discussing

Demo

Front End

- Html5
  - Define site layout with logo, header, footer, and main body content.
- CSS
  - Style the site layout based on CSS classes and ids
- Bootstrap
  - Make a nicely designed site quickly grids.
- Sass and the asset pipeline
  - Eliminate duplication in CSS .
- Rails
  - Define custom routing rules.

User Modelling

Generating a User model: `$ rails generate model User name:string email:string`

users	
id	integer
name	string
email	string
created_at	datetime
updated_at	datetime

Active Record comes with a host of methods for creating, saving, and finding data objects, all without having to use the structured query language used by relational databases.

User Modelling

User validations

- Validating Presence
  - validates :name, presence: true
  - validates :email, presence: true
- Validating Length
  - validates :name, presence: true, length: { maximum: 50 }
  - validates :email, presence: true, length: { maximum: 255 }
- Validating Format
  - VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
- Validating Uniqueness
  - uniqueness: true

Adding Secure Password

- Hashed Password
- Built-in **has_secure_password** method.
- Minimum Length Standards

users	
id	integer
name	string
email	string
created_at	datetime
updated_at	datetime
password_digest	string

Sign Up

- SignUp Form
  - Use form_for method(takes in an Active Record object and builds a form using the object's attributes.)
- Unsuccessful Signups
  - Display the error messages in the browser
- Successful Signups
  - Save the user,written to the database automatically
  - Redirect the browser to show the user's profile
- Flash
  - A message that appears on the subsequent page
  - Disappears upon visiting a second page or on page reload

Login

- Login
  - maintain state from one page to the next using temporary cookies via the session method.
  - Change features such as links on the layouts based on login status.
- Logout
  - using a destroy action to delete sessions
- Remember Users
  - Remember token and digest
  - Forgetting users
  - check box

Password Reset

- Password Resets Controller
  - Rails generate controller PasswordResets new edit --no-test-framework
- New passwords reset
  - Define the data model
- Password reset create action
  - Make a nicely designed site quickly grids.
- Password reset emails
  - Password reset mailer and templates

- Model
  - id(integer), content(text), user_id(integer), created_at(datetime), updated_time(datetime)
- Validation
  - validates :user_id, presence: true
  - validates :content, presence: true, length: { maximum: 140 }
- Relationship
  - Micropost belongs_to User
  - User has_many Micropost
- Optimization
  - default_scope -> { order(created_at: :desc)}
  - has_many :microposts, dependent: :destroy

- Controller
  - Require user logging in before creating or destroying his/her own microposts.
  - Create: POST
  - Destroy: DELETE
  - Active Record: feed.paginate(page: params[:page]), object counts
  - Error messages
  - Insert picture: CarrierWave, Picture Uploader, ImageMagick

following users function

generate the relation model

```
$ rails generate model Relationship follower_id:integer followed_id:integer
```

relationships	
id	integer
follower_id	integer
followed_id	integer
created_at	datetime
updated_at	datetime

- active relationships
- positive relationships

following users function

active relationship models (users.following)

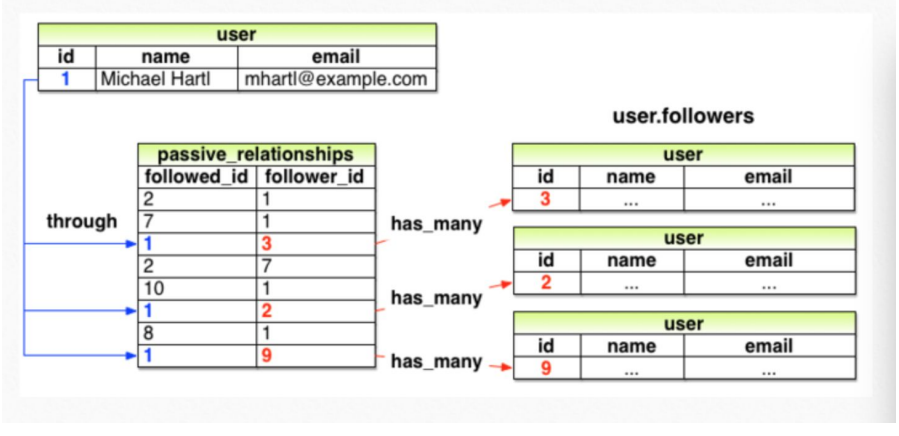
- Relationship
  - has_many :active_relationships
  - has_many :followeds, through: :active_relationships
  - belongs_to :follower/followed, class_name: "User"
- Validation
  - test follower_id, followed_id,
  - user.following.include?(other_user)
    user.following.find(other_user)
  - following.include?(other_user)

following users function

Method	Purpose
`active_relationship.follower`	Returns the follower
`active_relationship.followed`	Returns the followed user
`user.active_relationships.create(followed_id: other_user.id)`	Creates an active relationship associated with `user`
`user.active_relationships.create! (followed_id: other_user.id)`	Creates an active relationship associated with `user` (exception on failure)
`user.active_relationships.build(followed_id: other_user.id)`	Returns a new Relationship object associated with `user`

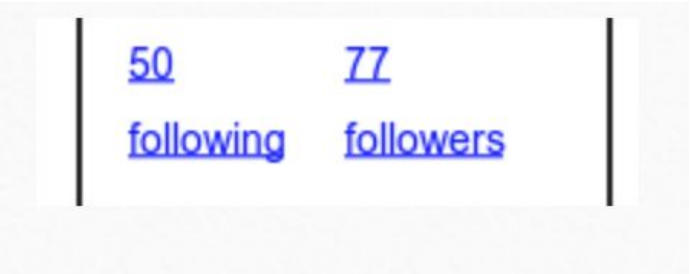
positive relationship models (users.followers)

- Relationship
  - has_many :passive_relationships, class_name: "Relationship"
  - has_many :followers, through: :passive_relationships, source: :follower
- Validation
  - assert archer.followers.include?(michael)
  - following.include?(other_user)

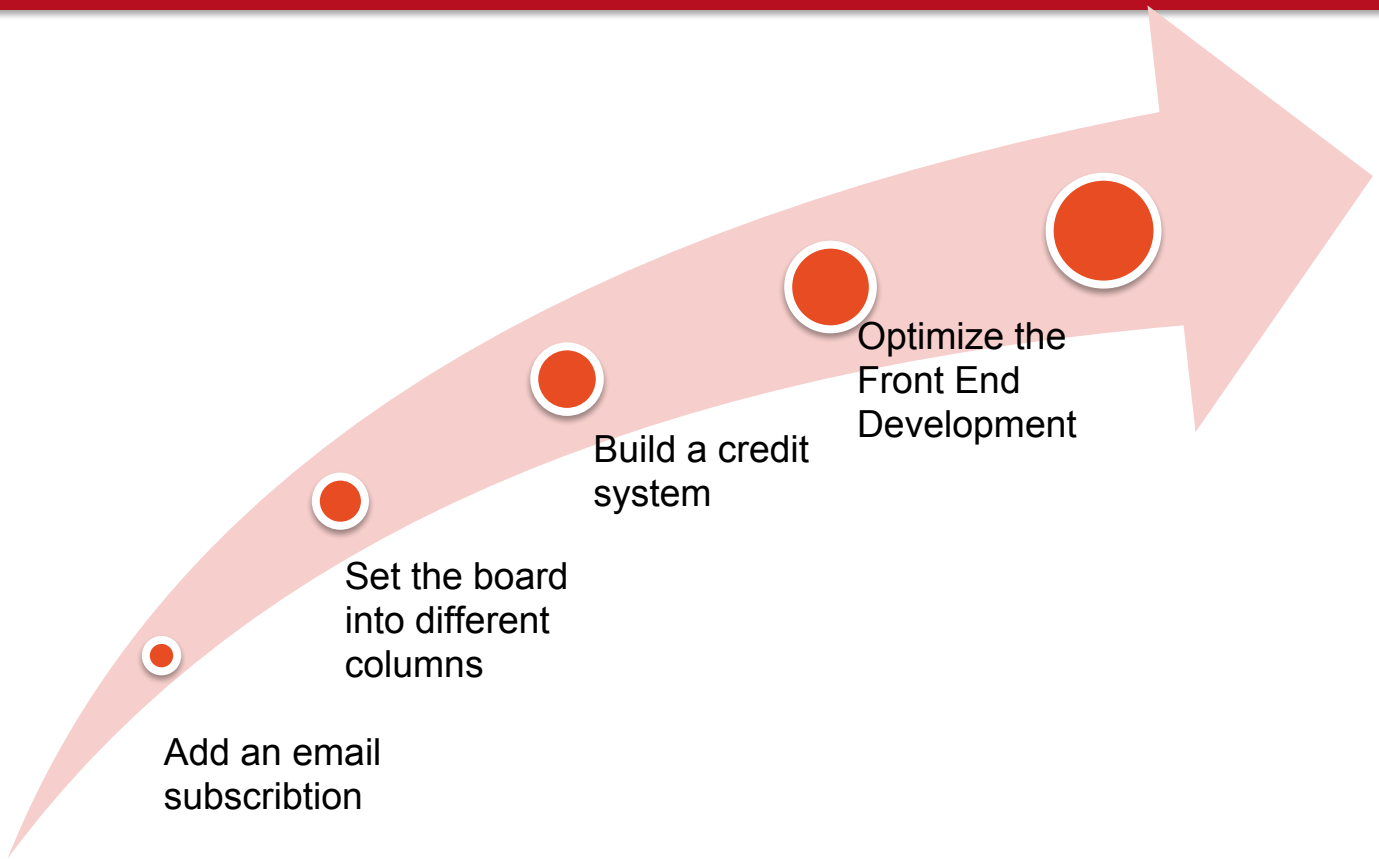


web interface for following users

- fill the database
  - use rails db:seed
  - following.each { |followed| user.follow(followed) }
    followers.each { |follower| follower.follow(user) }
- Stats and a following form
  - add following and follower actions to user controller (config.rb)
  - display user stats: *app/views/shared/_stats.html.erb*
  - add follower stats to home page: *app/views/static_pages/home.html.erb*
- following and follower pages
- working following button with Ajax



FurtherMore



THE OHIO STATE UNIVERSITY

Any Questions?