# Midterm 1 Review

Adnan Hemani

# Administrivia

- HW6 extends to 3/11 at 11:59 pm.
- No HW Party this week.

# Agenda

- REST / Routing
- SaaS Architecture
- Agile Methodology
- Velocity
- User Stories and BDD
- (if time) ActiveRecord

# REST / Routing

# RESTful API

API that uses HTTP requests such as GET, PUT, POST, DELETE, etc

# What are APIs?

- API =  Application Programming Interface
- A set of subroutine definitions, protocols, and tools for building software and applications
- Good APIs, you ask for? Here are examples:
  - Libraries and frameworks (sqrt(), sum(), print())
  - OS-Level API (fcntrl, etc.) - if you are triggered, I apologize
  - ...and hopefully your web API

# Examples of Twitter APIs

Public API

- The Search API
- The Search API: Tweets by Place
- Working with Timelines
- API Rate Limits
- API Rate Limits: Chart
- GET statuses/mentions_timeline
- GET statuses/user_timeline
- GET statuses/home_timeline
- GET statuses/retweets_of_me
- GET statuses/retweets/:id
- GET statuses/show/:id
- POST statuses/destroy/:id
- POST statuses/update
- POST statuses/retweet/:id
- POST statuses/unretweet/:id
- POST statuses/update_with_media
- GET statuses/oembed
- GET statuses/retweeters/ids
- GET statuses/lookup
- GET direct_messages/sent
- GET direct_messages/show
- GET search/tweets
- GET direct_messages
- POST direct_messages/destroy
- POST direct_messages/new
- GET friendships/no_retweets/ids

And many more….

# RESTful API

- In fact, you can build most web applications using GET.
  - Bad practice.
  - Ex. GET /login?username=saas&password=omgpwned
- Tips:
  - For read-only operation, use GET
  - Otherwise, use POST
  - Both GET and POST can pass parameters in URL
  - Additionally POST can pass parameters in the its packets

# More API Designs!

- GET, POST, PUT, DELETE
  - Not all browsers supports PUT and DELETE method in HTTP.
  - Both GET and POST can pass parameters in URL
  - Additionally POST can pass parameters in the its packets

# URL (Uniform Resource Locator)

https://www.etsy.com:443/search?q=test%20search#copy

- https://: protocol, others include http, ftp, etc.
- etsy : hostname, resolves to an IP address
- 443 : port number, 80 is standard for http
- /search: relative path
- q=test%20search: query terms, params
- copy: anchor, not technically part of request

# Design RESTful API

- Decide what resource(s) to be available
  - order, customer

# Design RESTful API

- Decide what resource(s) to be available
  - order, customer
- Assign URLs to those resources
  - /orders /customers

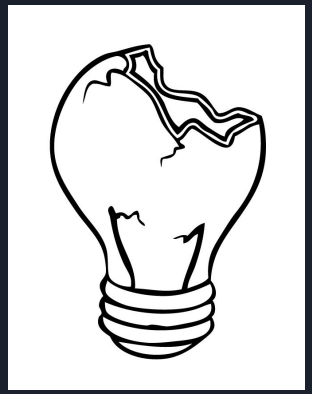
# Design RESTful API

- Decide what resource(s) to be available
  - order, customer
- Assign URLs to those resources
  - /orders /customers
- Decide what actions the client should be allowed to perform on those resources
  - GET /orders # list existing orders
  - POST /order # place a new order
  - GET /order/:id # get details for order :id

# Design RESTful API

- Decide what resource(s) to be available
  - order, customer
- Assign URLs to those resources
  - /orders /customers
- Decide what actions the client should be allowed to perform on those resources
  - GET /orders # list existing orders
  - POST /order # place a new order
  - GET /order/:id # get details for order :id
- Decide what pieces of data are required for each action and what format they should be in
  - POST /orders
  - data: {“crust”: “thin”, “toppings”: [“cheese”]}

# RESTful API Design Conventions



GET /getTodos
GET /getTodobyId/1
GET /updateTodo
GET /createTodo
GET /deleteTodo/1
...

GET todos
GET todos/1
POST todos/update
POST todos/create
POST todos/delete
...

# SaaS Architecture and SOA

# SaaS and SOA

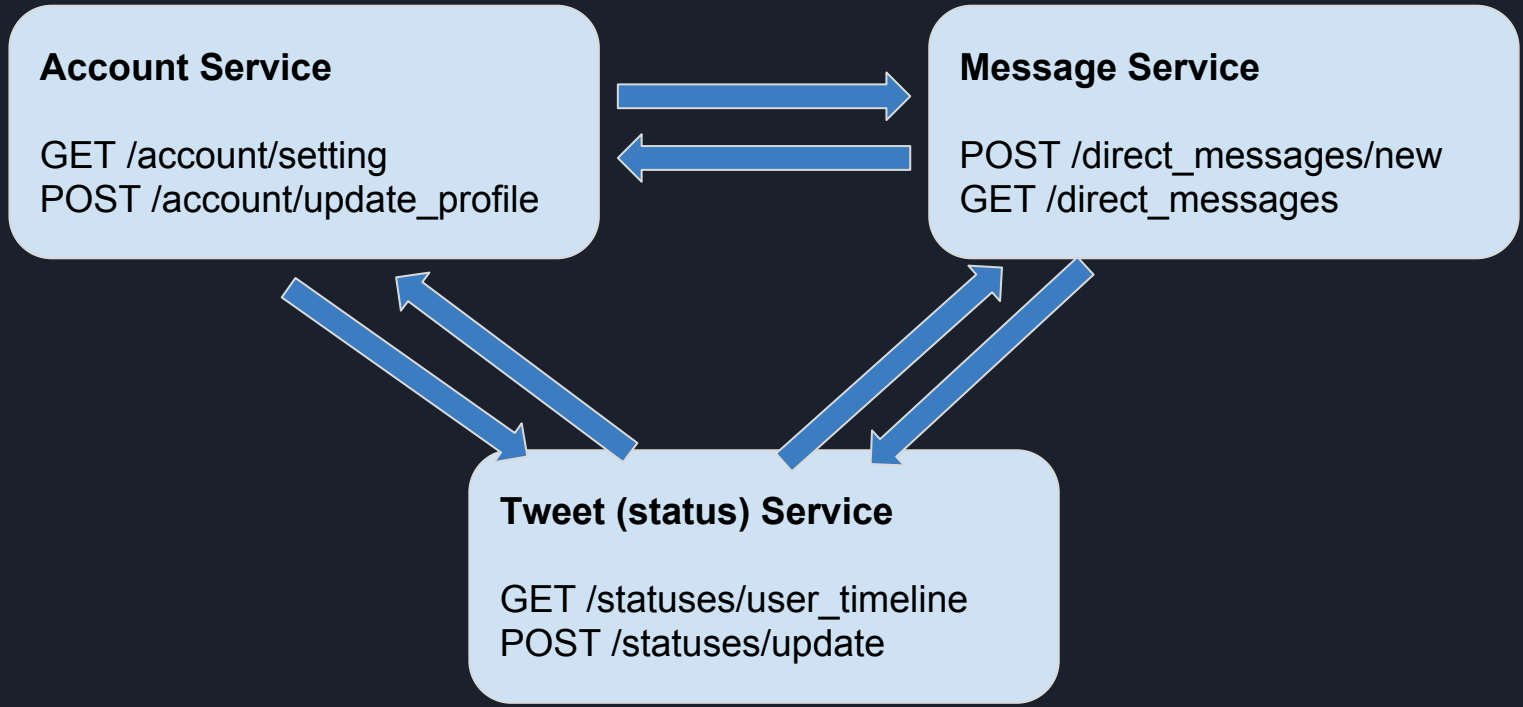
SaaS (Software as a Service):

- SaaS is just a method of software delivery
- Deliver software as (web) service instead of CD.
- You can do most of your IT tasks by using a browser.

SOA (Service Oriented Architecture):

- SOA is an architecture style to build software
- You can use SOA to build your SaaS application.
- A service is a program that can be interacted with through well-defined message exchanges
- SOA differs from the more general client/server model in its definitive emphasis on loose coupling between software components, and in its use of separately standing interfaces.
    - typically encapsulate a high-level business concept.
    - Service talk through web APIs (HTML, JSON, or XML).
- SOAs are like snowflakes – no two are alike.

# Twitter in SOA example



# Also remember that SaaS...

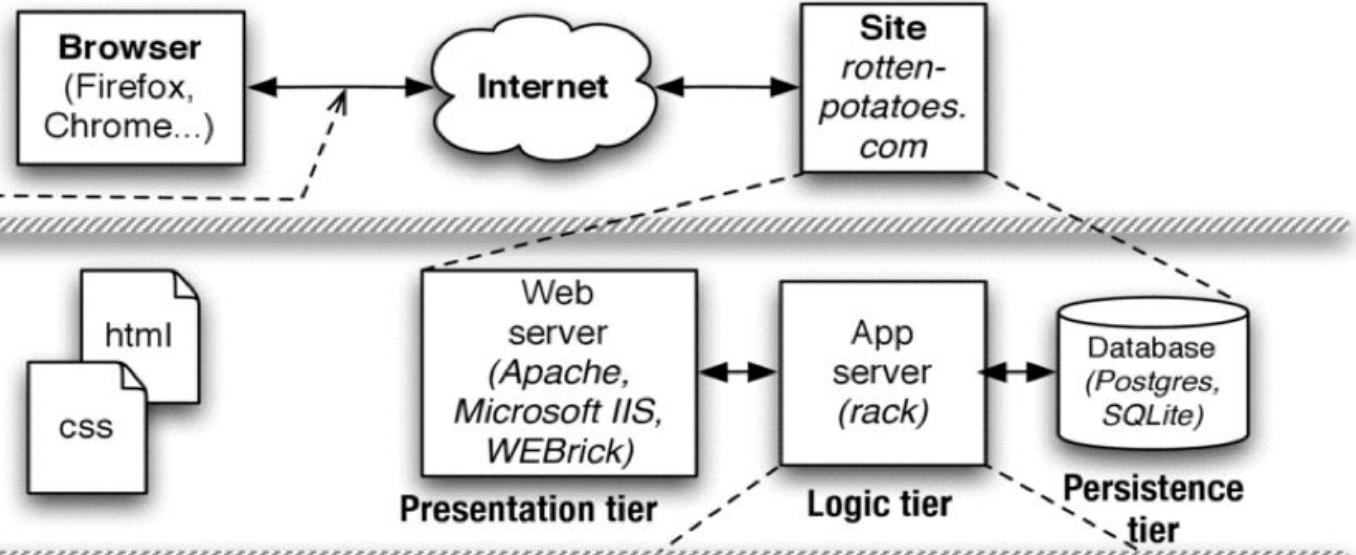
- Does communication using the HTTP/HTTPS protocol
  - HTTP(S) is stateless - what does this mean for us?
- Both works in pull and push
  - Pull: Receiving emails
  - Push: Receiving push notifications

**§2.1** 100,000 feet
- Client-server (vs. P2P)

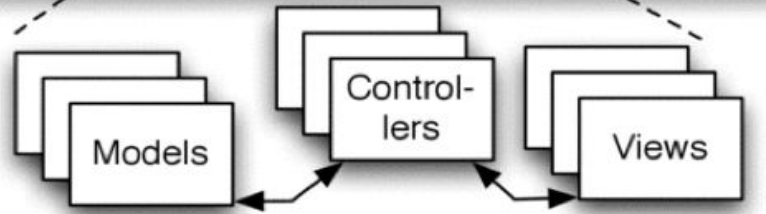
**§2.2** 50,000 feet
- HTTP & URIs

**§2.3** 10,000 feet
- XHTML & CSS

**§2.4** 5,000 feet
- 3-tier architecture
- Horizontal scaling




**§2.5** 1,000 feet—Model-View-Controller
(vs. Page Controller, Front Controller)




**§2.6** 500 feet: Active Record models (vs. Data Mapper)
**§2.7** 500 feet: RESTful controllers (Representational State Transfer for self-contained actions)
**§2.8** 500 feet: Template View (vs. Transform View)

- **Active Record**
- Data Mapper
- **REST**
- **Template View**
- Transform View

# Web Programming

CD was the main way of delivering software

- Network was small and slow
- Not many web users
- Web business logic was simple
- One server could handle all the requests

# Web Programming

- Web service is the main way of delivering software
- Network is big and fast
- Millions/billions of web users
- Web business logic can be super complex
- One server can no longer handle all the requests

# Methodologies

# First Came...Plan-and-Document

- Before coding, project manager makes plan
- Write detailed documentation all phases of plan
- Progress measured against the plan
- Changes to project must be reflected in documentation and possibly to plan
- First development process: Waterfall
  1. Requirements analysis & specification
  2. Architectural design
  3. Implementation & Integration
  4. Verification
  5. Operation & Maintenance
- Why? Easier to catch bugs earlier; documentation was great for new people

# Did it Work?

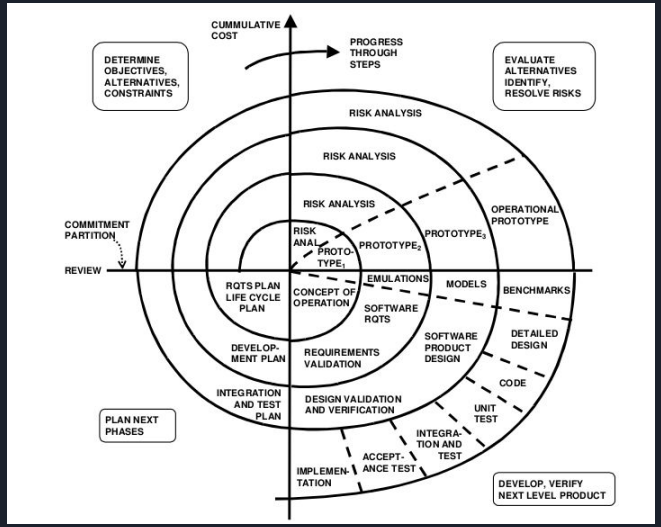
No.

# Why Not?!

- Was unable to adapt and change.
- These are called “top down” approaches

# Then came...the Spiral Lifecycle

- Use prototypes to get customer feedback until “final” version built
  - Iterations may be far apart
  - New prototype delivered every iteration



# And Finally Came...Agile

- Embraces change as a fact of life: continuous improvement vs. phases
- Developers continuously refine working but incomplete prototype until customers happy, with customer feedback on short Iterations (1-2 weeks)
- All lifecycle elements in every iteration
- Agile emphasizes Test-Driven Development (TDD) to reduce mistakes, written-down User Stories to validate customer requirements, Velocity to measure progress

# Velocity

# Cost Estimation

- Use velocity for this
- Can you use user stories as a unit for cost estimation?

# Velocity

- You should assign each story some points relating to its difficulty
- Velocity = avg. points per week
- Can you compare velocities across teams?

# User Stories and BDD

# BDD / Cucumber

- Emphasizes working closely with stakeholders, especially to avoid miscommunication
- User stories capture app behavior (document user requirements)
  - Written as a couple sentences on 3x5 index cards
  - All stakeholders brainstorm and prioritize features
- Tests behavior, NOT implementation
  - Even if implementation changes, ensures behavior stays the same
- Use Cucumber to implement different scenarios (use cases) that can occur under each user story

# Be SMART?

- Specific
- Measurable
- Achievable
- Relevant
- Timeboxed

# User Stories => Acceptance Tests!

- User tests:
  - Feature name
  - As a [kind of stakeholder],
  - So that [I can achieve some goal],
  - I want to [do some task]
- Acceptance Tests, use these keywords instead:
  - Given, When, Then, And, But
  - Regex will be used to turn these into tests

# Acceptance Tests

- These are NOT code - they look like normal English!
- Our way of doing acceptance tests: Cucumber
- Used alongside Capybara, which is a fake user that simulates a browser.

# ActiveRecord

We have gotten very lucky…

I should go to Vegas

# ActiveRecord (This is not a fitness app)

- An Implementation of the object-relational mapping (ORM) pattern.
- Automated mapping between classes and tables, attributes and columns
  - Basic operations on object: CRUD
  - (Create, Read, Update, Delete)
- Associations between objects defined by simple class methods (will be covered later)

# Example

```
class Article < ActiveRecord::Base {
  :id      => :integer,
  :title   => :string,
  :content => :text
}
```

AR automatically handles the mapping between:

- objects in memory
- Records in database

| id | title | content |
|---|---|---|
| 1 | First record | Hello world |
| 2 | Week 3 section | Active record etc. |
| 3 | Week 4 section | Rails etc. |

# Example

```
class Article < ActiveRecord::Base {
  :id      => :integer,
  :title   => :string,
  :content => :text
}
```

AR automatically handles the mapping between:

- objects in memory
- Records in database

```
a = Article.new
a.title = "Week 5"
a.save
```

| id | title | content |
|---|---|---|
| 1 | First record | Hello world |
| 2 | Week 3 section | Active record etc. |
| 3 | Week 4 section | Rails etc. |
| 4 | Week 5 | |

# Example

```
class Article < ActiveRecord::Base {
  :id      => :integer,
  :title   => :string,
  :content => :text
}
```

AR automatically handles the mapping between:

- objects in memory
- Records in database

```
Article.create(
  :title => "Week 5"
)
```

| id | title | content |
|---|---|---|
| 1 | First record | Hello world |
| 2 | Week 3 section | Active record etc. |
| 3 | Week 4 section | Rails etc. |
| 4 | Week 5 | |

# ActiveRecord & SQL

AR automatically handles the mapping between:

- objects in memory
- Records in database

AR will translate the query API call to SQL commands:

```
Article.where(:title => "Week 5")
> select * from Article where title = "Week 5"
```

The result will be put into objects in memory.

| id | title | content |
|---|---|---|
| 1 | First record | Hello world |
| 2 | Week 3 section | Active record etc. |
| 3 | Week 4 section | Rails etc. |
| 4 | Week 5 | |