

okmercury

Blatantly plagiarizing the okcupid matching algorithm to help people discover who to meet at a tradeshow.

WHO ARE WE

Cedric Hurst & Gary Turovsky

50% of Spantree Technology Group, LLC

Groovy, Grails, Java, Solr, Elasticsearch, Drools,
Backbone, Coffeescript, Hadoop


Planning Systems, Search Engine Design, Algorithms

Work across several industries with companies of
many sizes, focused exclusively on open source

WHAT'S OUR IDEA?

GIZMODO TOP STORIES

MATCH PERCENTAGE




A video player interface showing a 94% match percentage. On the left is a cartoon drawing of a woman with blonde hair, and on the right is a cartoon drawing of a man with a beard. A large play button is centered over the '94%' text.

DATING

Here's How OkCupid Uses Math to Find Your Match

FEB 14, 2013 9:32 AM 15,387 60


Share +1 Like 70

 **Leslie Horn**

Everyone you know has an online dating profile and if they say they don't they are lying to you. We can poke fun at it all we want, but there's actually a [mathematical formula](#) behind the digital match-making.

In this video, one of OkCupid's founders, Christian Rudder, explains how his site's algorithm works. When two people join the site, of course their shared interests are taken into account. These internet romance mathematicians look at that info as data, which they crunch through some equations to hopefully find you someone you'll like. Sure attraction is abstract, but as far as OkCupid is concerned, either your next random hookup or your lifelong match could be found by running some numbers. [[TED-Ed](#)]

WHAT'S OUR IDEA?



NEW A-List feature: See who rated you highly! Upgrade to A-List »

Help Topics

Topics Match Percentages Send Us Feedback

Calculating Match Percentages

What exactly those numbers mean.

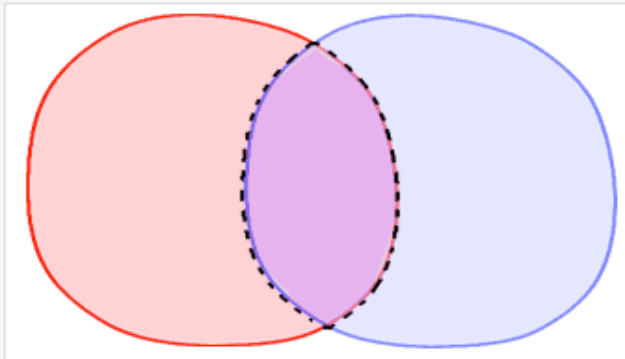
This is a brief, but technical, explanation of how your match percentages are calculated. It's a little complicated, but our method is quite interesting—even unique. Also, there's a patent pending, so no funny business.

Lets get started

We start wanting to calculate a **match percentage** for you and someone else. And we want to avoid mistakes at all costs! We collect three values for all users. When you answer a question on our [Improve Matches](#) page, we learn:

1. your **answer**,
2. how you'd like **someone else** to answer, and
3. how **important** the question is to you.

Your match percentage with a given person on OkCupid, let's call him B, is based on the values of (1), (2), and (3) for questions you've *both* answered. We'll call that set S later in this explanation:



Questions You Answered

MYTHOLOGY LESSON

Mercury was the patron god of financial gain, commerce, eloquence, messages/communication, travelers, boundaries and luck.

HOW IT WORKS

Users submit questions they'd like answered by other attendees



The screenshot shows a web interface for 'okmercury'. At the top left is the 'okmercury' logo, and at the top right is a 'Logout gary' button. Below the logo is the heading 'Create your question'. Underneath is a section titled 'Your match question' containing a large text input field with the placeholder text 'I think test-driven development is...'. Below this is a section titled 'Possible answers' with five input fields. The first three fields contain the text 'Essential', 'Probably a good idea', and 'A Waste of time|'. The last two fields are labeled 'Possible answer 4' and 'Possible answer 5'. At the bottom of the form are two buttons: 'Add Another Question >' and 'Done Adding Questions'.

okmercury Logout gary

Create your question

Your match question

I think test-driven development is...

Possible answers

Essential

Probably a good idea

A Waste of time|

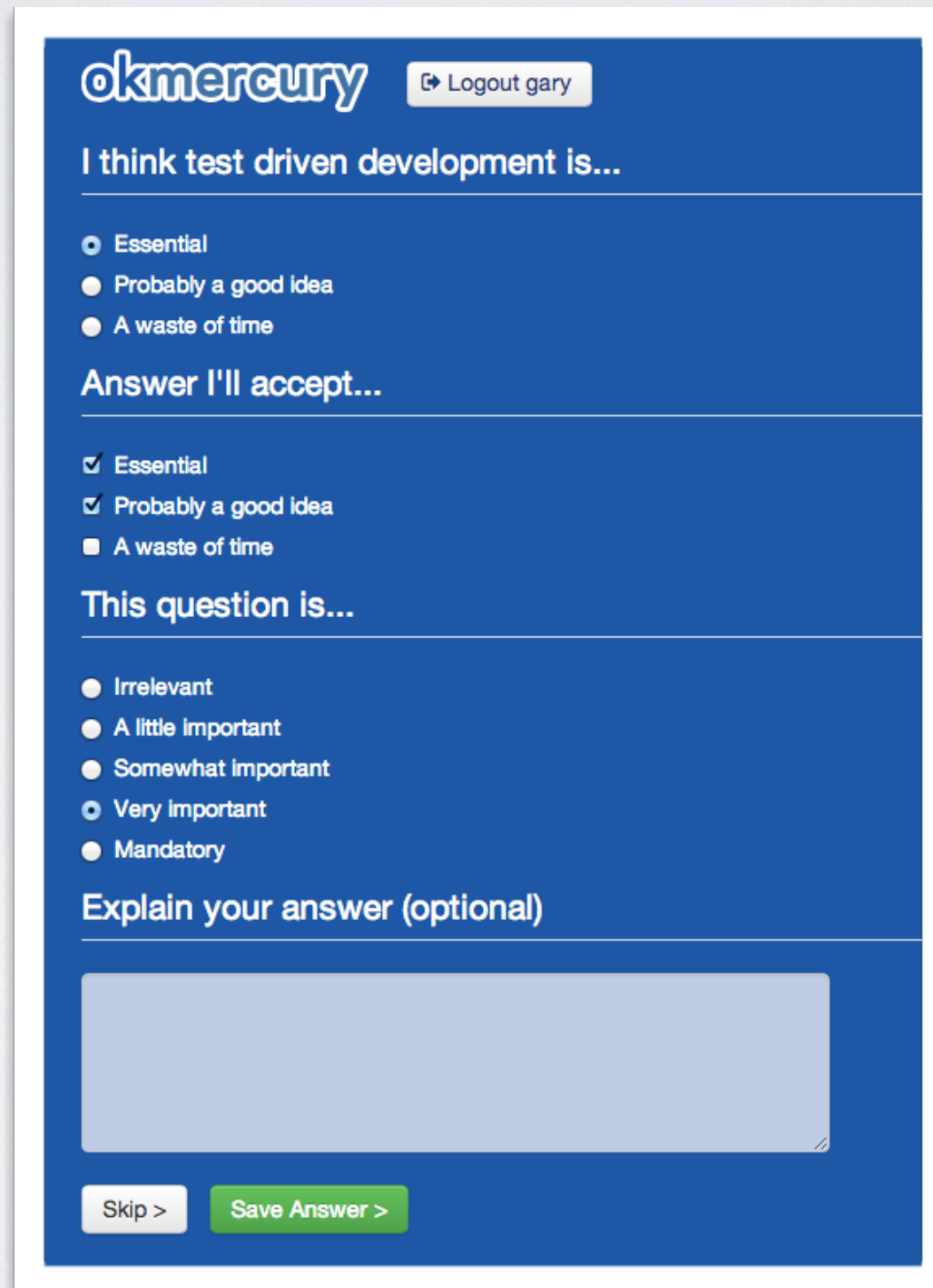
Possible answer 4

Possible answer 5

Add Another Question > Done Adding Questions

HOW IT WORKS

Other attendees answer the question for themselves and specify possible answers for the people they'd like to meet



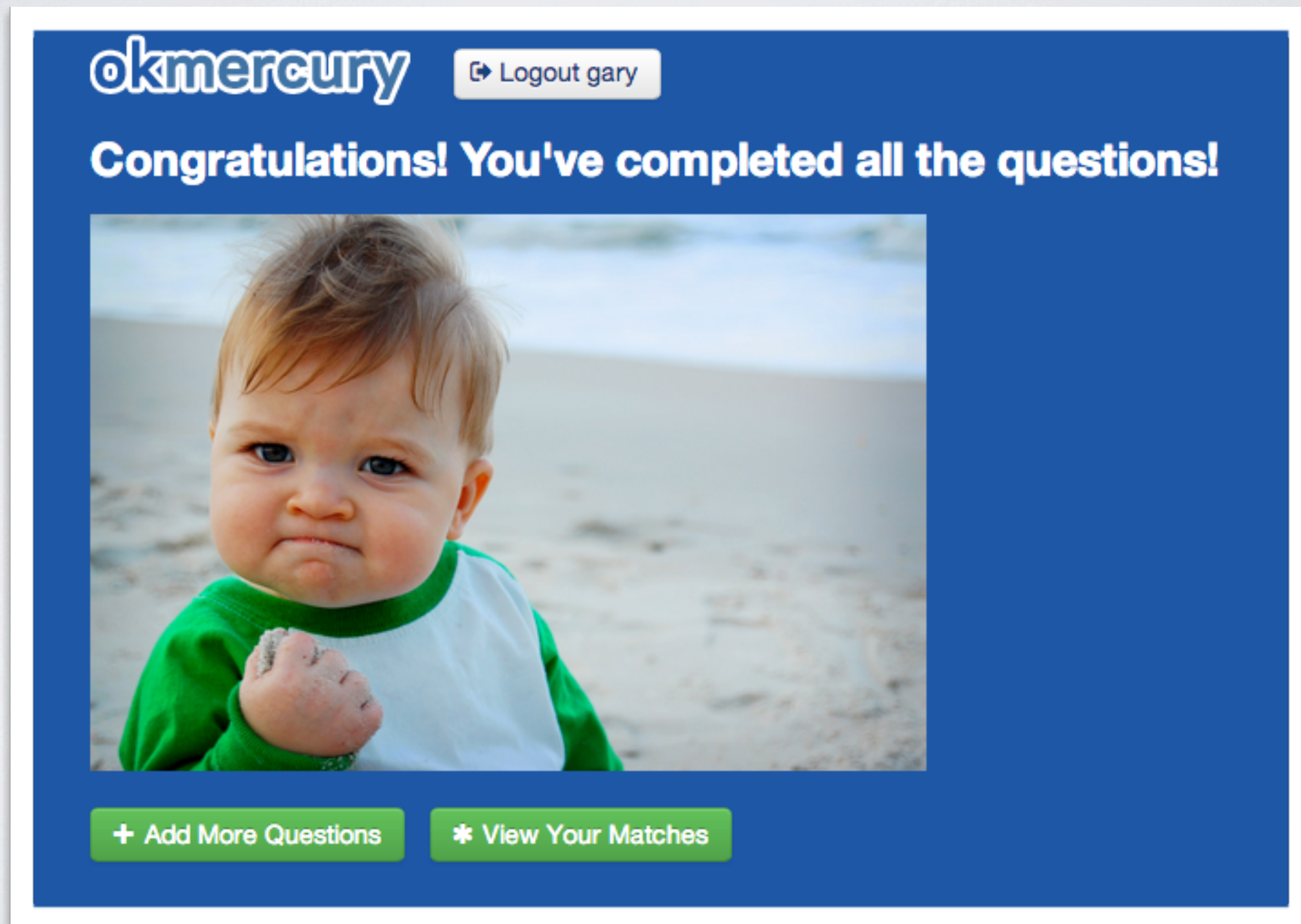
The screenshot shows a web application interface for 'okmercury'. At the top, there is a logo and a 'Logout gary' button. The main content area is divided into four sections, each with a title and a list of options:

- I think test driven development is...**
 - ☒ Essential
 - ☐ Probably a good idea
 - ☐ A waste of time
- Answer I'll accept...**
 - ☒ Essential
 - ☒ Probably a good idea
 - ☐ A waste of time
- This question is...**
 - ☐ Irrelevant
 - ☐ A little important
 - ☐ Somewhat important
 - ☒ Very important
 - ☐ Mandatory
- Explain your answer (optional)**
 - A large text input area for providing an explanation.

At the bottom, there are two buttons: 'Skip >' and 'Save Answer >'.


HOW IT WORKS

Once an attendee is finished answering questions, they can add more or view their matches



HOW IT WORKS




We do do a bunch of math and recommend other attendees to meet at the event



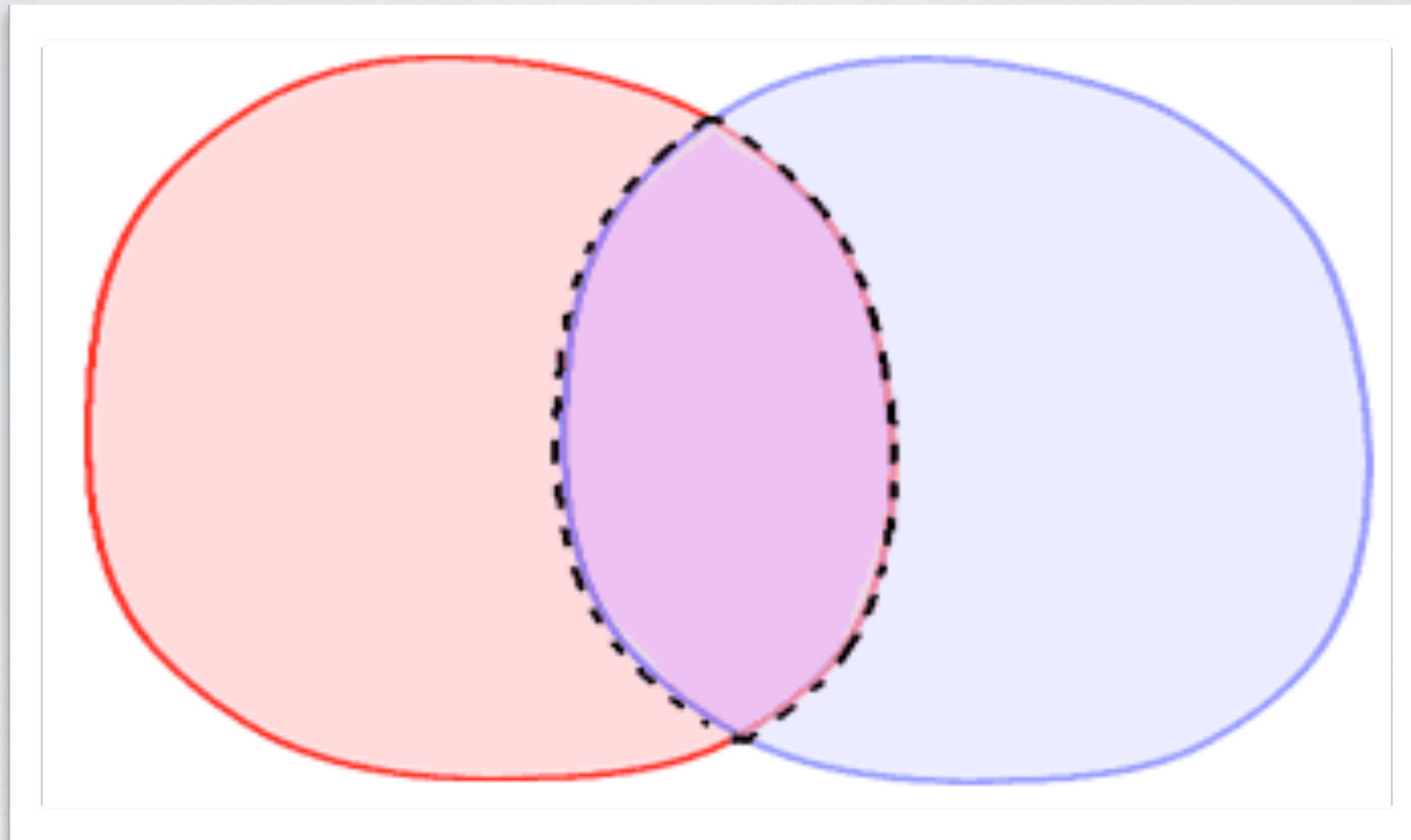
[Logout cedric](#)

We think you should meet Gary (86%)!

People You Should Meet

Name	Overall Score	Their Score for You	Your Score for Them	Questions in Common
 Gary Turovsky	85.71%	100.00%	74.29%	7
 Jane Smith	62.06%	52.38%	73.53%	6
 John Doe	56.34%	95.24%	33.33%	3

THE MATH



Questions You Answered
Questions You Both Answered (S)
Questions B Answered

THE MATH

Important data...

(A) Your answer

(B) How you'd like someone else to answer

(C) How important the question is to you

IMPORTANCE

Weighted based on how much you care

Irrelevant	0
A little important	1
Somewhat important	10
Very important	50
Mandatory	250

THE MATH

How much did John's answer make you happy?

If their answer is in your list of acceptable answers:

The importance score of how much you care

If their answer is not in your list of acceptable answers:

Not at all!

THE MATH

How much did your answer make John happy?

*If your answer is in their list of acceptable answers:
The importance score of how much they care*

*If your answer is not in their list of acceptable answers:
Not at all!*

THE MATH

Should you want to meet John?

Questions John got right
(weighted by your importance)

Total possible points for common questions
(also weighted by your importance)

THE MATH

Should John want to meet you?

Questions you got right
(weighted by John's importance)

Total possible points for common questions
(also weighted by John's importance)

THE MATH

Should you and John meet?

Square Root of (Your Score x John's Score)

THE STACK



THE DATA

Users

```
{
  "_id": { "$oid" : "512114BA690031FE535496DA" },
  "email": null,
  "firstName": "Cedric",
  "gravatarHash": null,
  "lastName": "Hurst",
  "passwordHash": "password",
  "roles": null,
  "version": 0
}
```

THE DATA

Questions

```
{
  "_id": { "$oid" : "51214F476900748871D48B8D" },
  "assignedId": { "$oid" : "51214F476900748871D48B8C" },
  "createdBy": { "$oid" : "5121156D690031FE535496DE" },
  "createdDate": { "$date": 1361137479000.000000 },
  "lastModifiedDate": { "$date": 1361137479000.000000 },
  "question": "I work for...",
  "userIdsThatHaveAnswered": [
    { "$oid" : "5121156D690031FE535496DE" },
    { "$oid" : "5121159C690031FE535496E4" },
    { "$oid" : "51211DD0690031FE535496F9" },
    { "$oid" : "5121541F69009FE41349DB76" },
    { "$oid" : "5121582E6900B162FE229765" }
  ],
  "version": 1
}
```


THE DATA

QuestionOptions

```
{  
  "_id": { "$oid" : "51214F476900748871D48B8E" },  
  "answer": "Myself",  
  "order": "1.0",  
  "question": { "$oid" : "51214F476900748871D48B8D" },  
  "version": 1  
}
```

THE DATA

Answers

```
{
  "_id": { "$oid" : "51214F706900748871D48BBE" },
  "acceptableAnswerIds": [
    { "$oid" : "51214F476900748871D48B9E" },
    { "$oid" : "51214F476900748871D48B9C" },
    { "$oid" : "51214F476900748871D48B9B" }
  ],
  "importance": "A_LITTLE_IMPORTANT",
  "lastModifiedDate": { "$date": 1361137520000.000000 },
  "question": { "$oid" : "51214F476900748871D48B9A" },
  "skipped": false,
  "user": { "$oid" : "5121156D690031FE535496DE" },
  "userAnswer": { "$oid" : "51214F476900748871D48B9D" },
  "userAnswerExplanation": "",
  "version": 0
}
```


THE DATA

QuestionMatches

```
{  
  "_id": { "$oid" : "512150EAC3820BDA0C3E7B67" },  
  "pointsPossibleForUserA": 10,  
  "pointsPossibleForUserB": 50,  
  "questionId": { "$oid" : "51214F476900748871D48BA7" },  
  "scoreForUserA": 10,  
  "scoreForUserB": 0,  
  "userId": { "$oid" : "5121156D690031FE535496DE" },  
  "userId": { "$oid" : "5121159C690031FE535496E4" }  
}
```

THE DATA

UserMatches

```
{
  "_id": { "$oid" : "51215112C3820BDA0C3E7B6F" },
  "matchPercentageScore": 0.941176,
  "matchPoints": 320,
  "matchPointsPossible": 340,
  "matchUserId": { "$oid" : "5121159C690031FE535496E4" },
  "overallScore": 0.833333,
  "principalPercentageScore": 0.900000,
  "principalPoints": 270,
  "principalPointsPossible": 300,
  "principalUserId": { "$oid" : "51211DD0690031FE535496F9" },
  "questionsInCommon": 6
}
```


PUTTING IT ALL TOGETHER

As users answer questions, we find other user's answers to those questions and calculate the score from both sides and “upsert” the QuestionMatch into MongoDB.

PUTTING IT ALL TOGETHER

```
def handleAnswer(ObjectId answerId) {
  Answer answer = Answer.get(answerId)
  ReentrantLock lock = getLock(questionHyperLock, questionLocks, answer.question.id)
  lock.lock()
  try {
    List<Answer> otherAnswers = getOtherUserAnswers(answer)
    otherAnswers.each { Answer otherAnswer ->
      Answer answerA, answerB
      if(answer.user.id < otherAnswer.user.id) {
        answerA = answer
        answerB = otherAnswer
      } else {
        answerA = otherAnswer
        answerB = answer
      }
      upsertQuestionMatch(answerA, answerB)
      updateUserMatch(answerA.user, answerB.user, answerA.user)
      updateUserMatch(answerB.user, answerA.user, answerA.user)
    }
  } finally {
    lock.unlock()
  }
}
```


PUTTING IT ALL TOGETHER

After we calculate the
QuestionMatch, we
(re)calculate the user match
using the MongoDB
Aggregation Framework and
“upsert” a UserMatch.

PUTTING IT ALL TOGETHER

```
void updateUserMatch(User principalUser, User matchUser, User userA) {
    String key = "${principalUser.id}->${matchUser.id}"
    ReentrantLock lock = getLock(userMatchHyperLock, userMatchLocks, key)
    lock.lock()
    try {
        User userB = (userA == principalUser ? matchUser : principalUser)

        DBObject match = [userId: userA.id, userBId: userB.id] as BasicDBObject

        log.info "Match: ${match}"

        DBObject group = [
            _id: [userId: '$userAId', userBId: '$userBId'],
            principalPoints: [sum: '$scoreForUserA'],
            principalPointsPossible: [sum: '$pointsPossibleForUserA'],
            matchPoints: [sum: '$scoreForUserB'],
            matchPointsPossible: [sum: '$pointsPossibleForUserB'],
            questionsInCommon: [sum: 1]
        ] as BasicDBObject

        AggregationOutput out = questionMatchCollection.aggregate([$match: match], [$group: group])

        Iterator<DBObject> resultsIterator = out.results().iterator()

        if(resultsIterator.hasNext()) {
            DBObject results = resultsIterator.next()

            Float marginOfError = getMarginOfError(results.questionsInCommon)
            Float principalPercentageScore = scorePercentage(results.principalPoints, results.principalPointsPossible)
            Float matchPercentageScore = scorePercentage(results.matchPoints, results.matchPointsPossible)
            Float overallScore = Math.min((1F-marginOfError), Math.sqrt(principalPercentageScore * matchPercentageScore))

            DBObject criteria = [principalUserId: principalUser.id, matchUserId: matchUser.id] as BasicDBObject

            DBObject update = [$set: [
                principalPoints: results.principalPoints,
                principalPointsPossible: results.principalPointsPossible,
                principalPercentageScore: principalPercentageScore,
                matchPoints: results.matchPoints,
                matchPointsPossible: results.matchPointsPossible,
                matchPercentageScore: matchPercentageScore,
                questionsInCommon: results.questionsInCommon,
                overallScore: overallScore
            ]] as BasicDBObject

            update['$set'].putAll(criteria)

            log.info "criteria: ${criteria}"

            userMatchCollection.update(criteria, update, true, false, WriteConcern.SAFE)
        } else {
            log.error "No aggregation results found for user match ${key}"
        }
    } finally {
        lock.unlock()
    }
}
```


PUTTING IT ALL TOGETHER

When an attendee wants to see their matches, we simply query the UserMatch collection sorting by `overallMatch` score.

PUTTING IT ALL TOGETHER

```
List<DBObject> getBestMatchesForUser(User user, String sortField = 'overallScore') {
    DBObject criteria = [principalUserId: user.id] as BasicDBObject
    DBObject sortMap = ["${sortField}": -1] as BasicDBObject
    return userMatchCollection.find(criteria).toArray()?.sort {
        println it
        Float v = it[sortField]
        return v != null ? -v : -Integer.MAX_VALUE
    }
}

DBObject getBestMatchForUser(User user) {
    List matches = getBestMatchesForUser(user)
    return matches ? matches[0] : null
}
```


COOL (GEEKY) STUFF WE DID

You can use it right now

<http://okmercury.co>

Completely RESTful API

Our frontend consumes it

100% Open-Source (Apache 2.0)

<http://github.com/Spantree/okmercury>

STUFF WE DIDN'T HAVE TIME FOR

Support user registration and OAuth

Allow attendees to edit previous questions

Add multi-tenant support

Use backbone for rich front-end

Create an attendee profile page

Do the number crunching with map-reduce

CHECK IT OUT

The screenshot shows the GitHub interface for the repository **Spantree / okmercury**. At the top, there's a navigation bar with links to Explore, Gist, Blog, and Help. The repository name is prominently displayed, along with buttons for Pull Request, Unwatch, Star (0), and Fork (0). Below this, a tabbed interface shows Code (selected), Network, Pull Requests (0), Issues (17), Wiki, Graphs, and Settings. The 'Code' tab displays cloning options: Clone in Mac, ZIP, HTTP, SSH, and Git Read-Only. The SSH URL is `git@github.com:Spantree/okmercury.git`. Below the cloning options, there's a section for the current branch, **develop**, with tabs for Files, Commits, and Branches (2). The **Files** tab is active, showing a list of files and their commit history. The latest commit is by **divideby0**, titled 'Update README.md', 4 minutes ago. The file list includes `docs`, `grails-app`, `puppet`, `src`, `web-app`, `.gitignore`, `README.md`, `Vagrantfile`, and `application.properties`.

PUBLIC **Spantree / okmercury** Pull Request Unwatch Star 0 Fork 0

Code Network Pull Requests 0 Issues 17 Wiki Graphs Settings

No description or homepage.

Clone in Mac ZIP HTTP SSH Git Read-Only `git@github.com:Spantree/okmercury.git` Read+Write access

branch: **develop** Files Commits Branches 2 Tags

okmercury / 105 commits

Update README.md

divideby0 authored 4 minutes ago latest commit 38a1f14bf0

docs	3 hours ago	Create MD_Questions [mdabney]
grails-app	2 hours ago	fixing syntax error [divideby0]
puppet	7 hours ago	#35: updating mongodb to 2.2.3 [divideby0]
src	3 hours ago	#37: refactoring matching algorithm [divideby0]
web-app	2 hours ago	miscellaneous fixes [divideby0]
.gitignore	a day ago	#11: adding .vagrant to gitignore [divideby0]
README.md	4 minutes ago	Update README.md [divideby0]
Vagrantfile	a day ago	#11: adding vagrant config [divideby0]
application.properties	a day ago	Removing hibernate line from properties file [flyhighplato]

<http://github.com/Spantree/okmercury>

SAY HELLO

Web: <http://www.spantree.net>

Twitter: [@spantreeinc](#) [@divideby0](#) [@flyhighplato](#)

Github: <http://www.github.com/Spantree>

LinkedIn: <http://www.linkedin.com/company/spantree-technology-group-llc>

Email: info@spantree.net