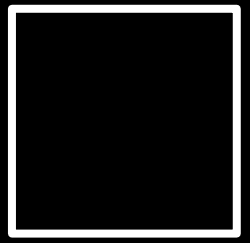
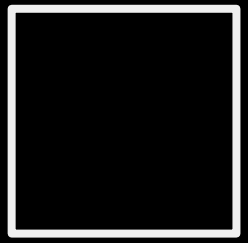
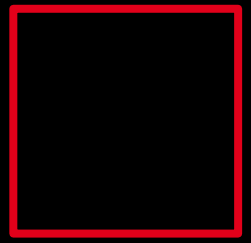
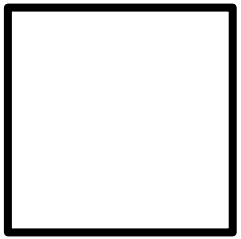
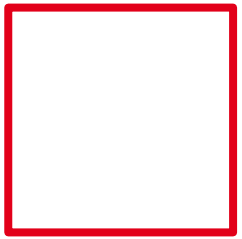




White Text in Red Box





Hello!

# About Me


- Torben Reetz
- 7th Semester Software Engineering (SSD)
- 3reetz@informatik.uni-hamburg.de

# A Recommender Framework for Skills Management

@SinnerSchrader

# Contents

- Motivation
- Requirements
- Commercial Solutions
- Concept
- Implementation
- Evaluation

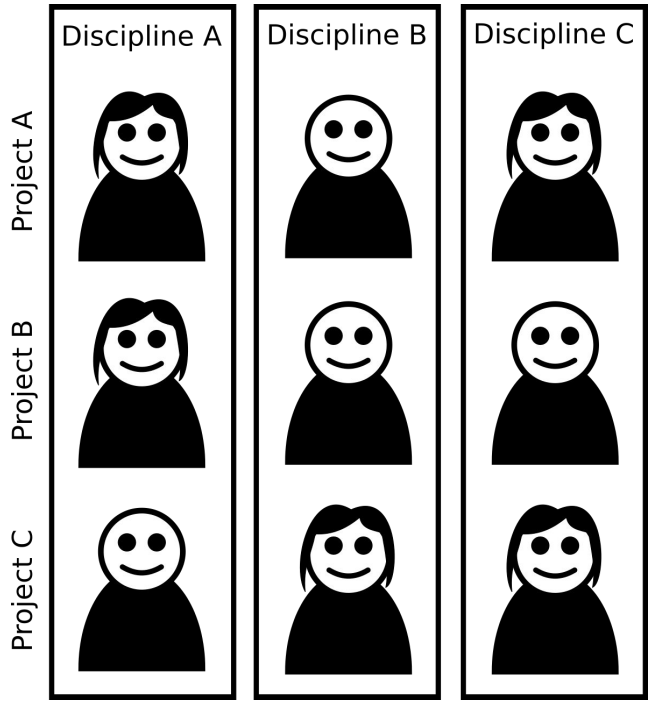
# Motivation

# Context SinnerSchrader

- Hamburg based
- Full service web agency
- 505 employees (459 full-time)
- Revenue > 51M Euro (15/16)

# Matrix Organization

- Domain specific teams
- Project teams
- Multiple project teams per employee
- Definitions:
  - Project Manager → Project team
  - Supervisor → Domain specific team



# Project Driven Business

- Employees work on multiple projects
- Limited in time
- Planning is hard
  - New requirements
  - New customers
  - Errors in estimation of workload

# Struggles (Project Manager)

- Employees leave
- Workload changes
  - new sub projects
- Shift in disciplines
  - Designers → Developers

Project managers frequently look for new team members.

# Struggles (Looking for Help)

- Different experience and knowledge
- Different fields to work in
  - Designers, Frontend Developers, Backend Developers, Writers, …
- Different projects need different competencies
  - Tech Stacks, Services ordered by the customer, …

Employees search for people that can help to solve a specific problem.

# Challenges to Tackle

- People search for other people that have specific skills
  - To solve a problem
  - To add them to a project team
- There is no central source of information

# Goal

- Create this source of information
- Focus: Motivation and Cooperation
  - Find someone who is motivated to work in the team/to help you
  - Working together ↔ Monitoring

# Requirements

# Functional Requirements

- Person search
  - Enter skills → find best matching person
- User profiles
  - Skills, personal data, direct contact
  - Enter own skills
  - Login
- Management of registered Skills
  - The system holds a pool of skills users can add to their profiles
  - Add new skills
  - Rename skills
  - Delete skills

# Non Functional Requirements

- Different devices
  - Primary: Desktops
  - Mobile devices optional
- Browser support
  - Chrome, Firefox, Safari
  - No support for IE/Edge
- Response Times (RAIL)
  - 100ms to acknowledge input
  - 1s to finish rendering results
- Scalability
  - Increased number of users should not be a problem
  - Enlarge storage and computing resources

# Commercial Solutions

# Skills Base

# Skills Base

# Talent Management (engage!)

# Talent Management (engage!)

# SkillsDB Pro

# SkillsDB Pro

# Many More

# Conclusion

# Concept

# Definitions

- Skill: ability a specific persons has
- Levels
  - Skill level: measurement of knowledge about a skill
  - Will level: measurement of motivation to employ a skill
  - Four Step Scale (0-3)
- Fitness: Measurement of how well a person fits into a searched skill set

# Four Step Scale

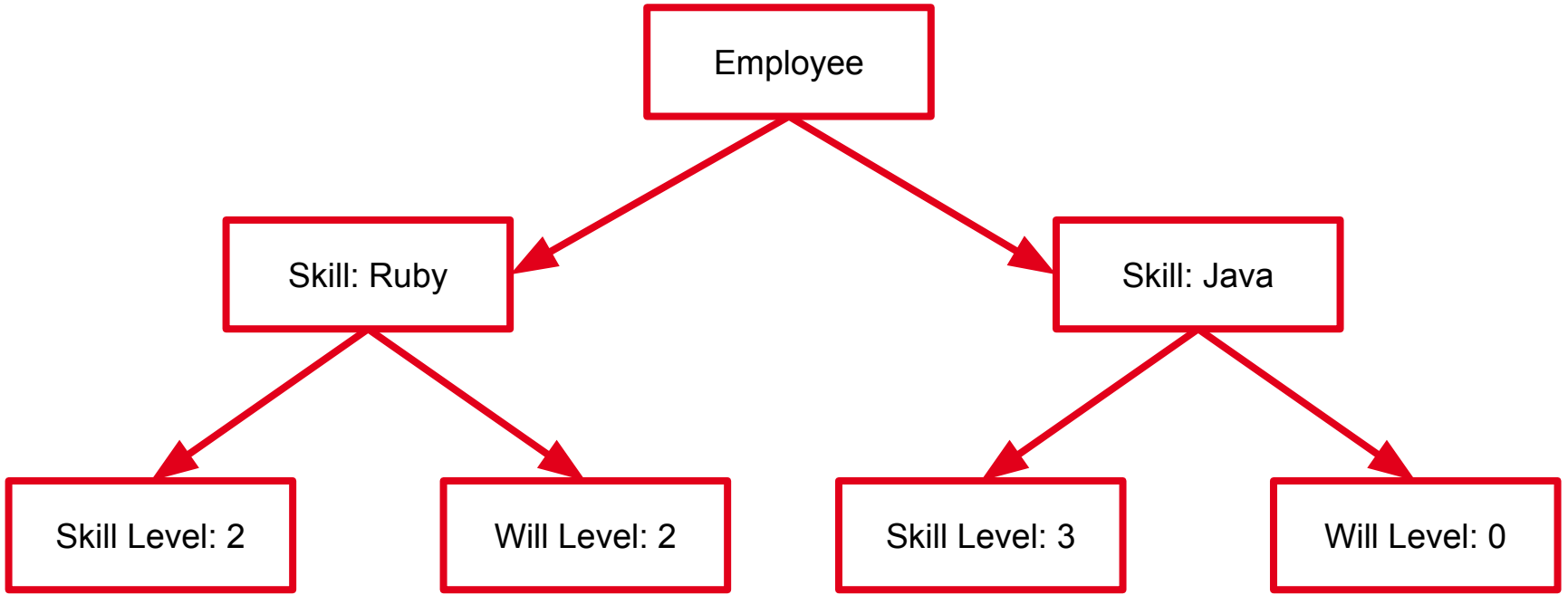
| | Skill Level | Will Level |
|---|---|---|
| 0 | Novice | Uninterested |
| 1 | Basic Knowledge | Indifferent |
| 2 | Advanced Knowledge | Somewhat Interested |
| 3 | Expert | Highly Interested |

# Four Step Scale

| | Skill Level | Will Level |
|---|---|---|
| 0 | Novice | Uninterested |
| 1 | Basic Knowledge | Indifferent |
| 2 | Advanced Knowledge | Somewhat Interested |
| 3 | Expert | Highly Interested |

- Skill Level = 0 → Person has little, but more than no knowledge
- No knowledge → Skill not present

# Definitions



# Person Search

1. User enters skills to look for
2. Systems presents list of results
    - People that have all skills
    - Best match on first Position
    - Recommender System

# Recommender Systems

“are information filtering systems that deal with the problem of information overload by filtering vital information fragment[s] out of large amount of [...] information”

I SINKAYE , F.O. ; F OLAJIMI , Y.O. ; O JOKOH , B.A.: Recommendation systems: Principles, methods and evaluation

# Recommender Systems

- Take a large amount of information
- Filter this information
- Recommend parts of it

# Recommender Systems

- Take a large amount of information → All employees
- Filter this information
- Recommend parts of it

# Recommender Systems

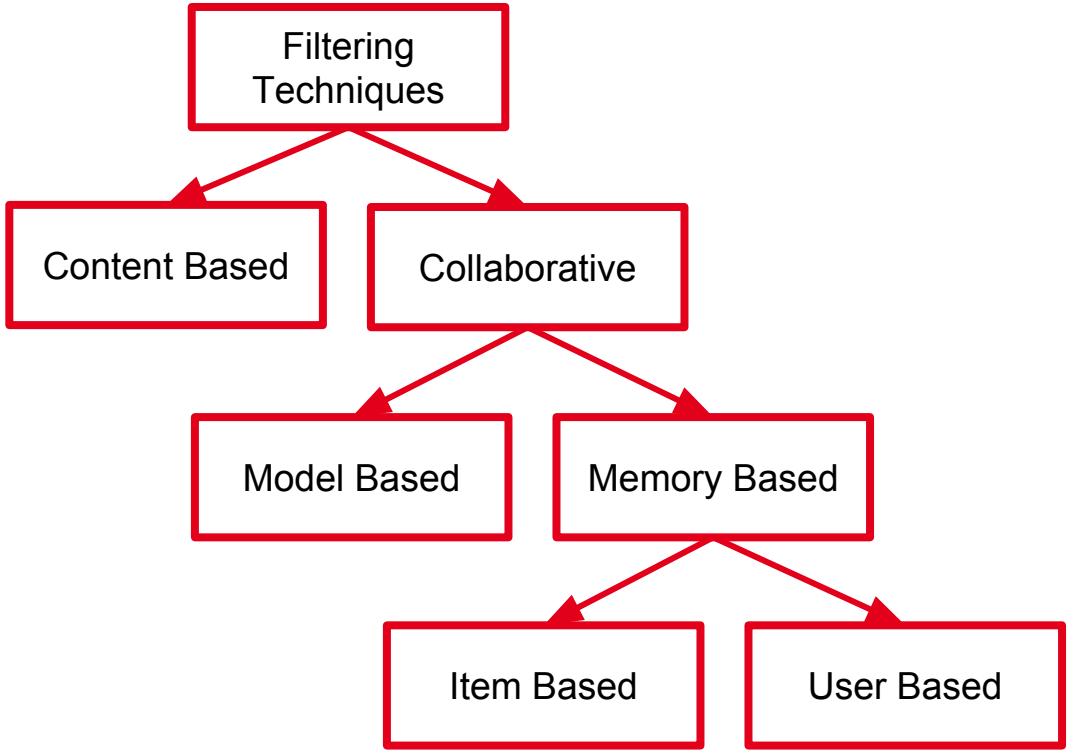
- Take a large amount of information → All employees
- Filter this information → Filtering by knowledge and motivation
- Recommend parts of it

# Recommender Systems

- Take a large amount of information → All employees
- Filter this information → Filtering by knowledge and motivation
- Recommend parts of it → Best match first

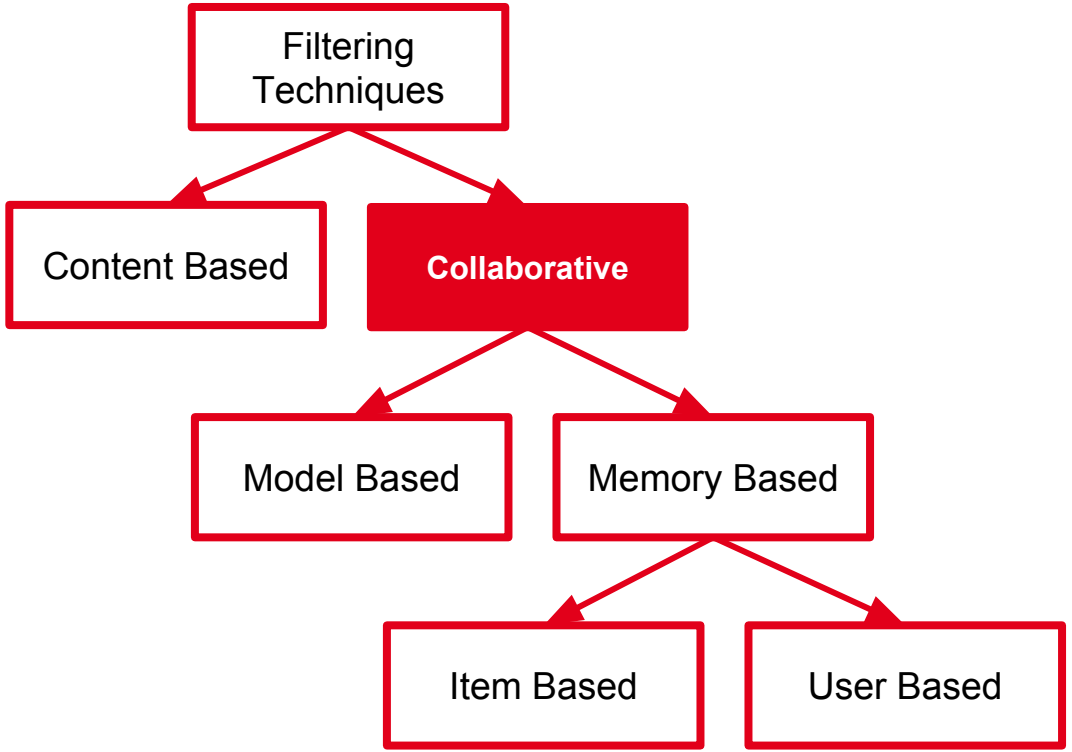
# Filtering Techniques

- Isinkaye et al.
- Techniques to find the items to suggest
- Hybrid filtering: combine multiple techniques



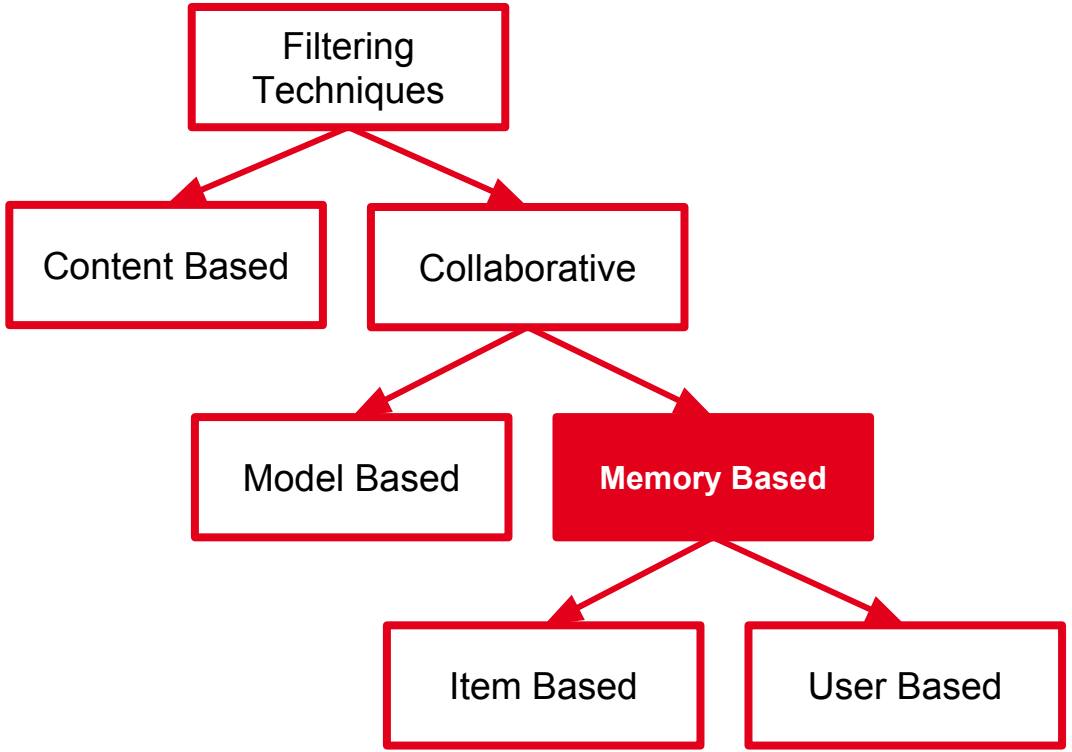
# Filtering Techniques (Collaborative)

- People behave similarly
- Find neighbours



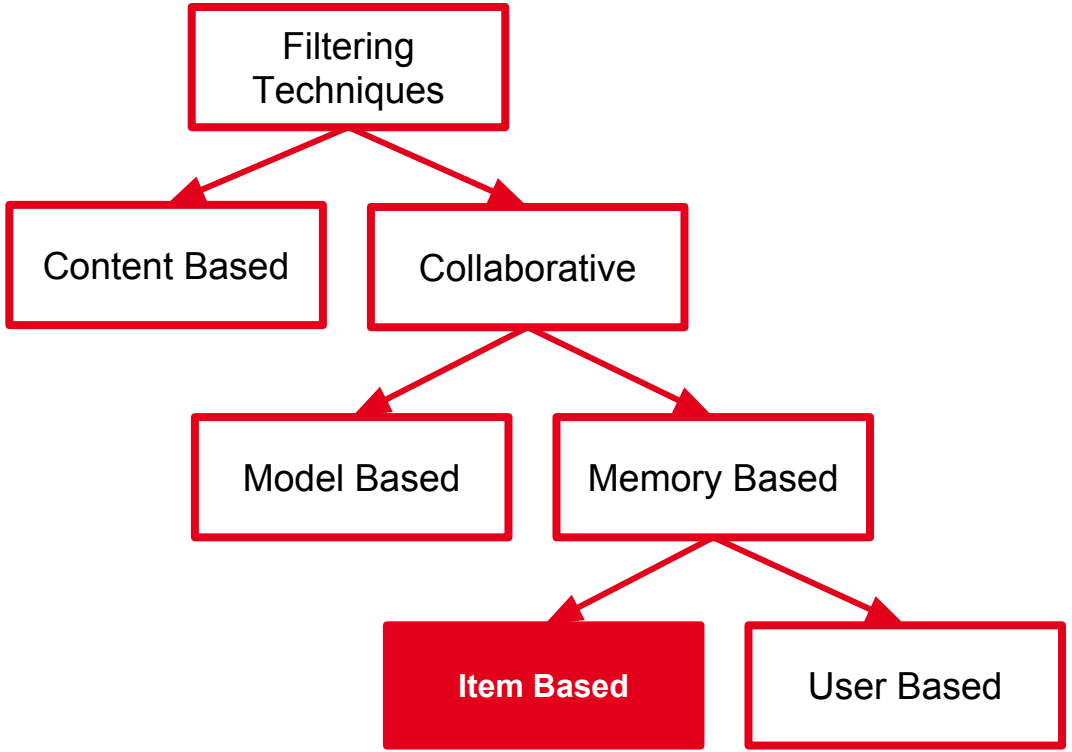
# Filtering Techniques (Memory Based)

- Subset of collaborative filtering
- Operates directly on saved interaction history



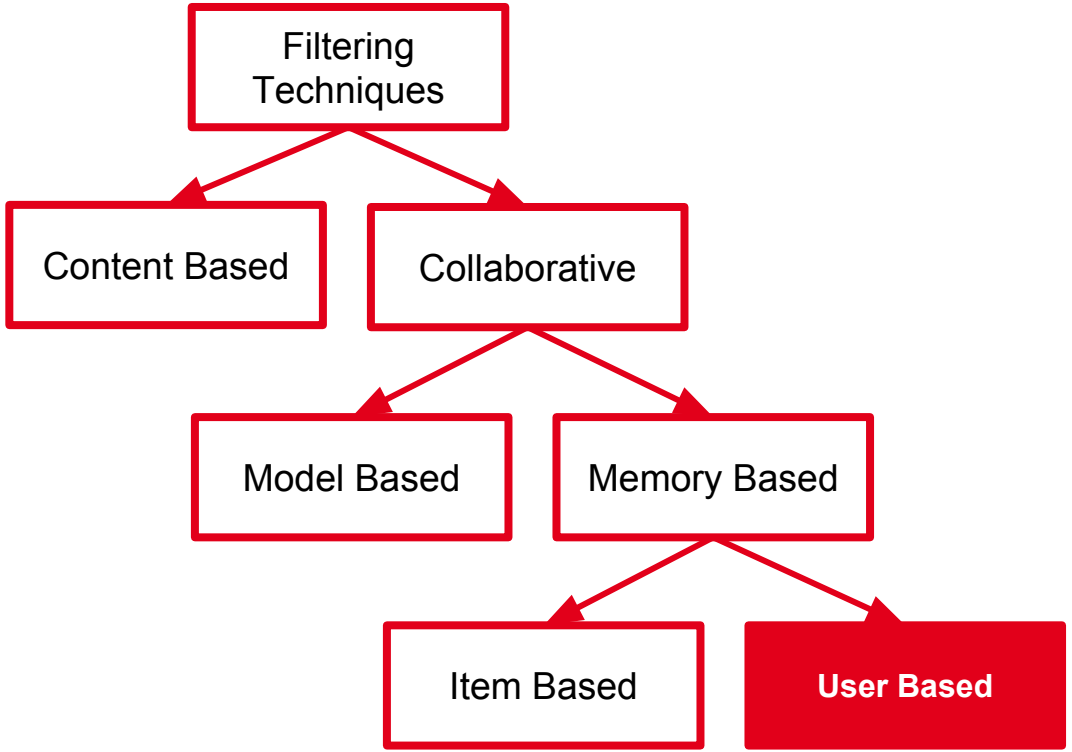
# Filtering Techniques (Item Based)

- Subset of memory based filtering
- Save interaction history per item
- Find items that are similar to each other



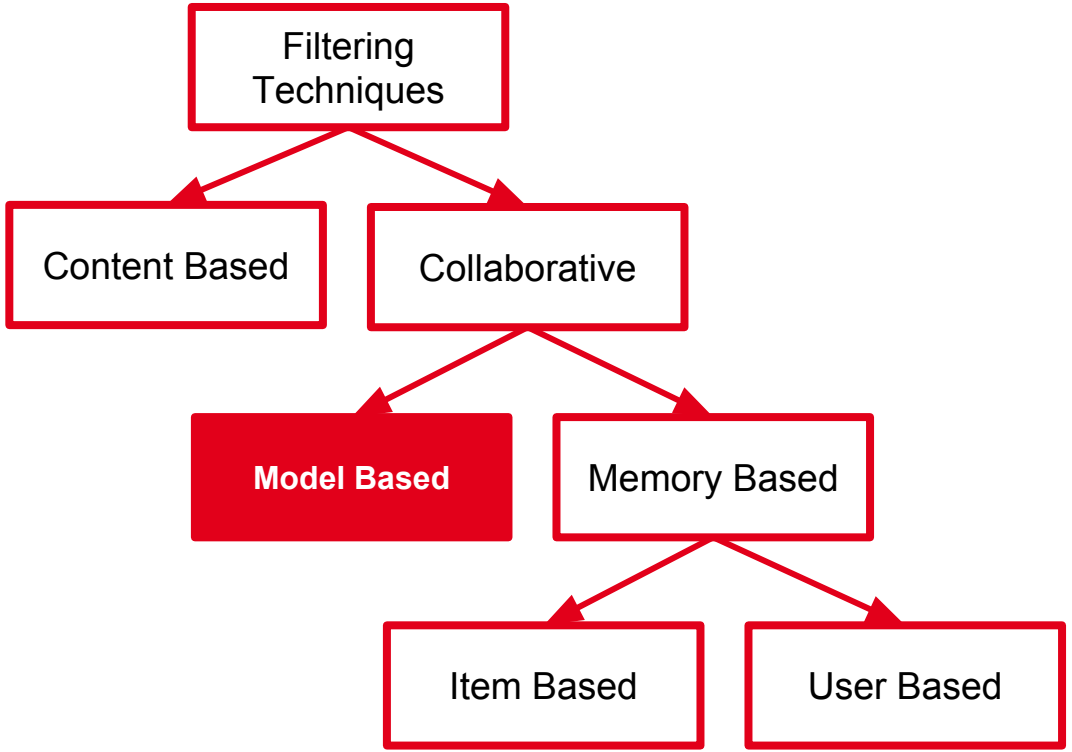
# Filtering Techniques (User Based)

- Subset of memory based filtering
- Save interaction history per user
- Find groups of users that have similar interaction histories



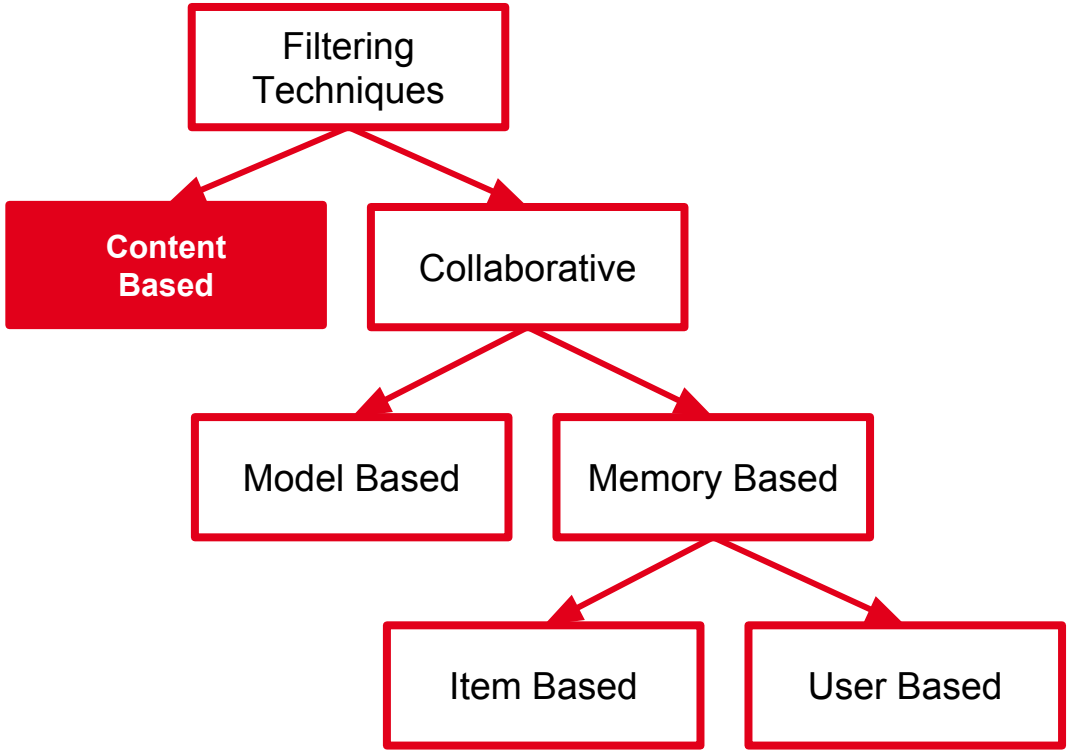
# Filtering Techniques (Model Based)

- Subset of collaborative filtering
- Model used to create suggestions
- Interactions to learn the model

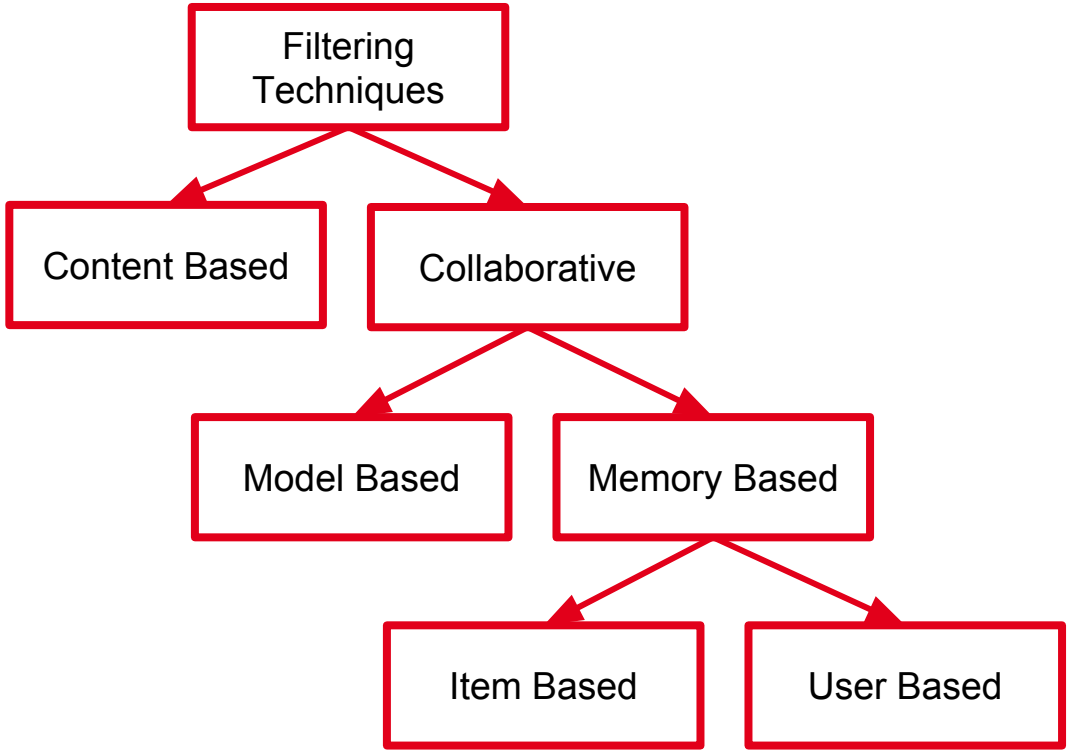


# Filtering Techniques (Content Based)

- Examine items, not interactions
- Reference items (previous interactions, searched)
- Find items that have attributes similar to reference items

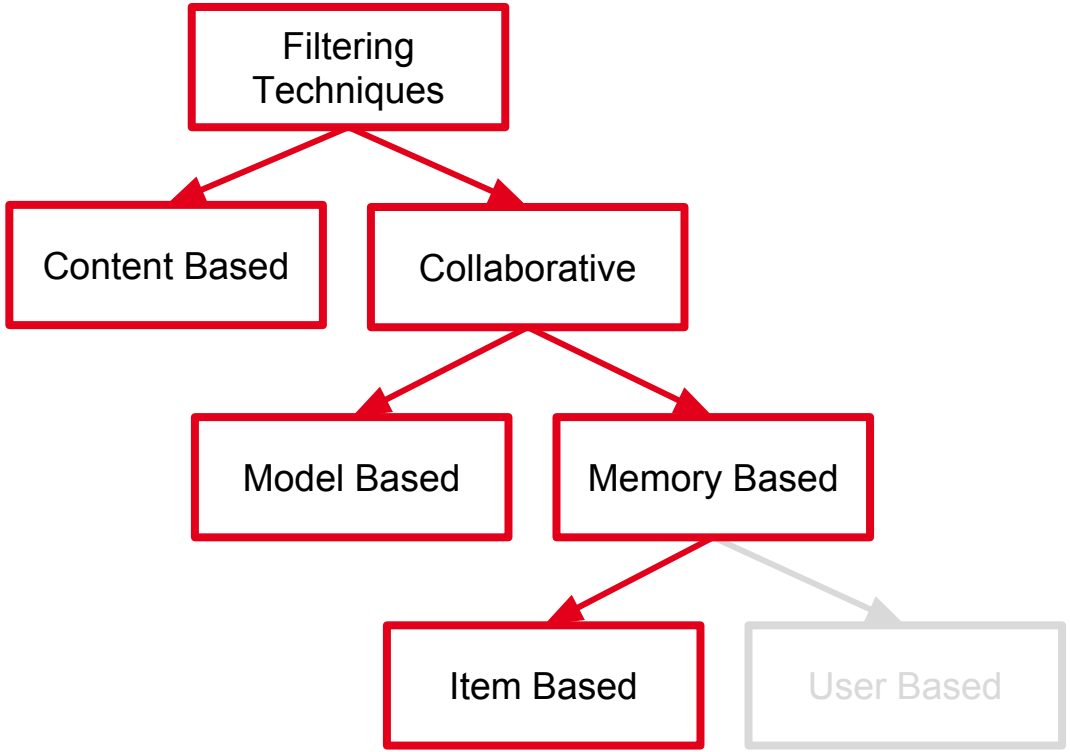


# Filtering Techniques (Chosen Approach)



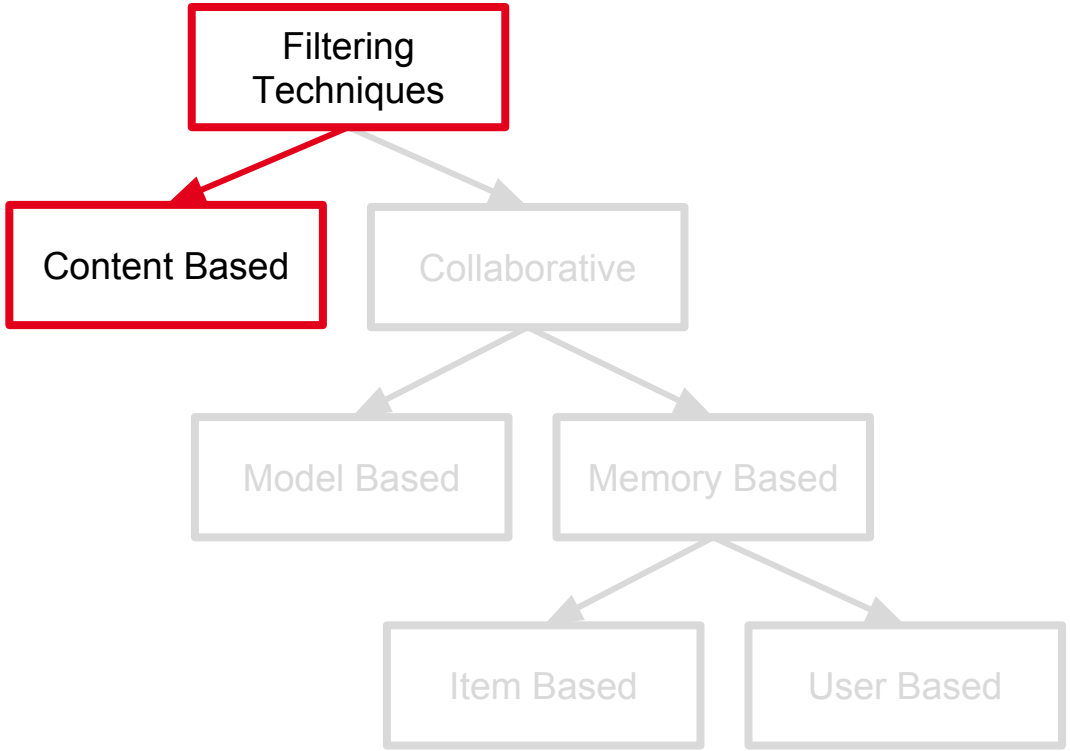
# Filtering Techniques (Chosen Approach)

- User not logged in



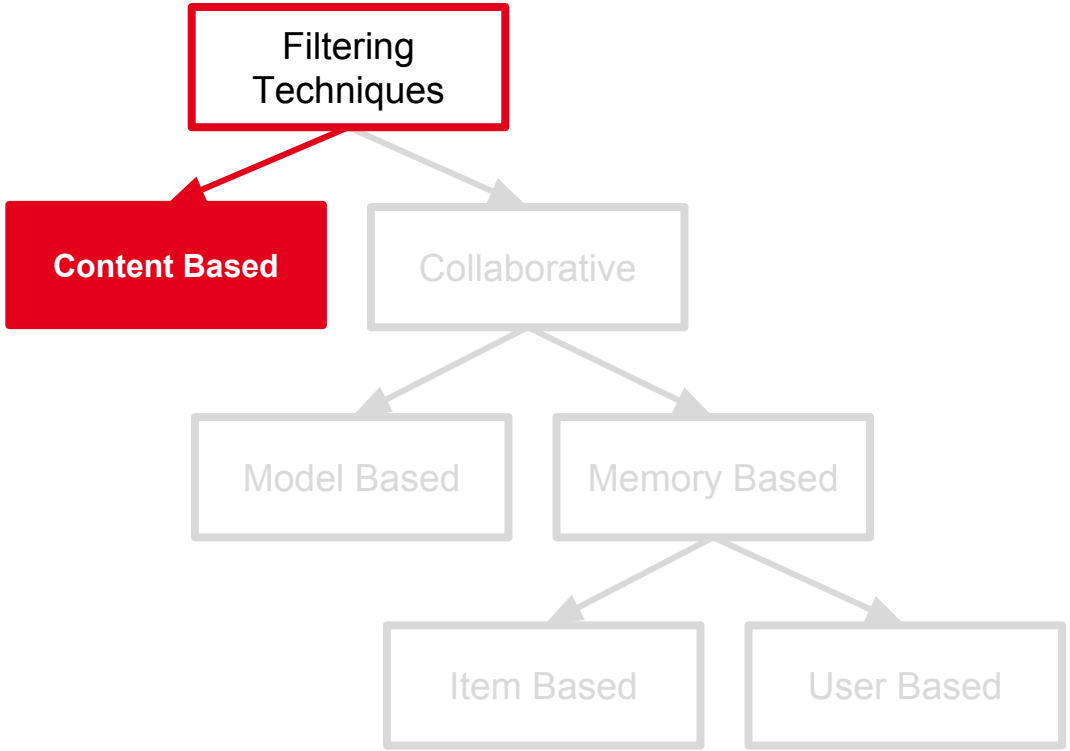
# Filtering Techniques (Chosen Approach)

- User not logged in
- Other users' searches must not affect the current one



# Filtering Techniques (Chosen Approach)

- User not logged in
- Other users' searches must not affect the current one
- Items (employees) have rich attributes
- Specific reference items (search query)
- Decision: Content Based Filtering



# Person Search (Outline)


- User already submitted their search query
- User entered a location to search at (optional)

# Person Search (Outline)


- Create a list of everyone known to the system

# Person Search (Outline)


- Remove persons who do not have all searched skills in their profile
- Levels are ignored at this time

# Person Search (Outline)


- User can restrict the search to locations
- If so, remove all employees not working there

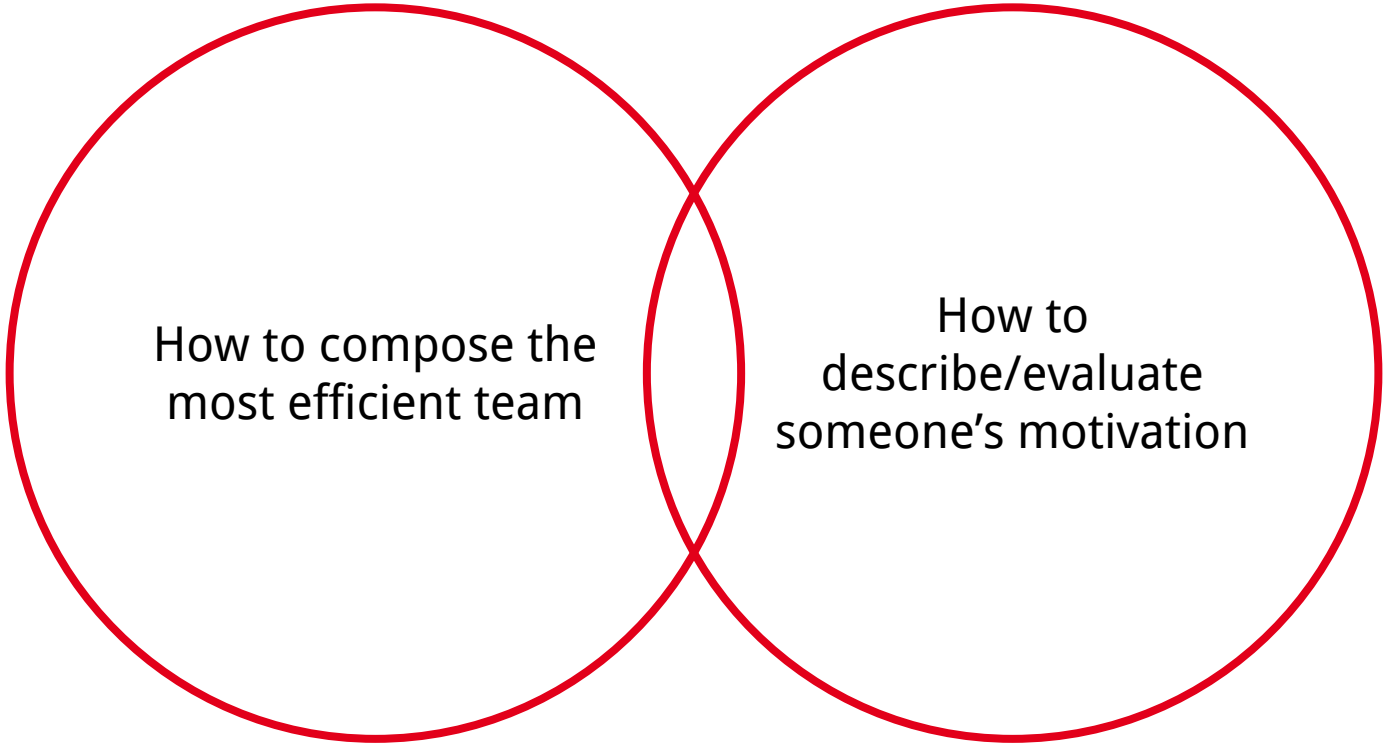
# Person Search (Outline)


- For every employee in the list, create a score that
  - Describes how well they fit the search criteria
  - Takes into account skill levels and will levels
  - On a scale from zero to one

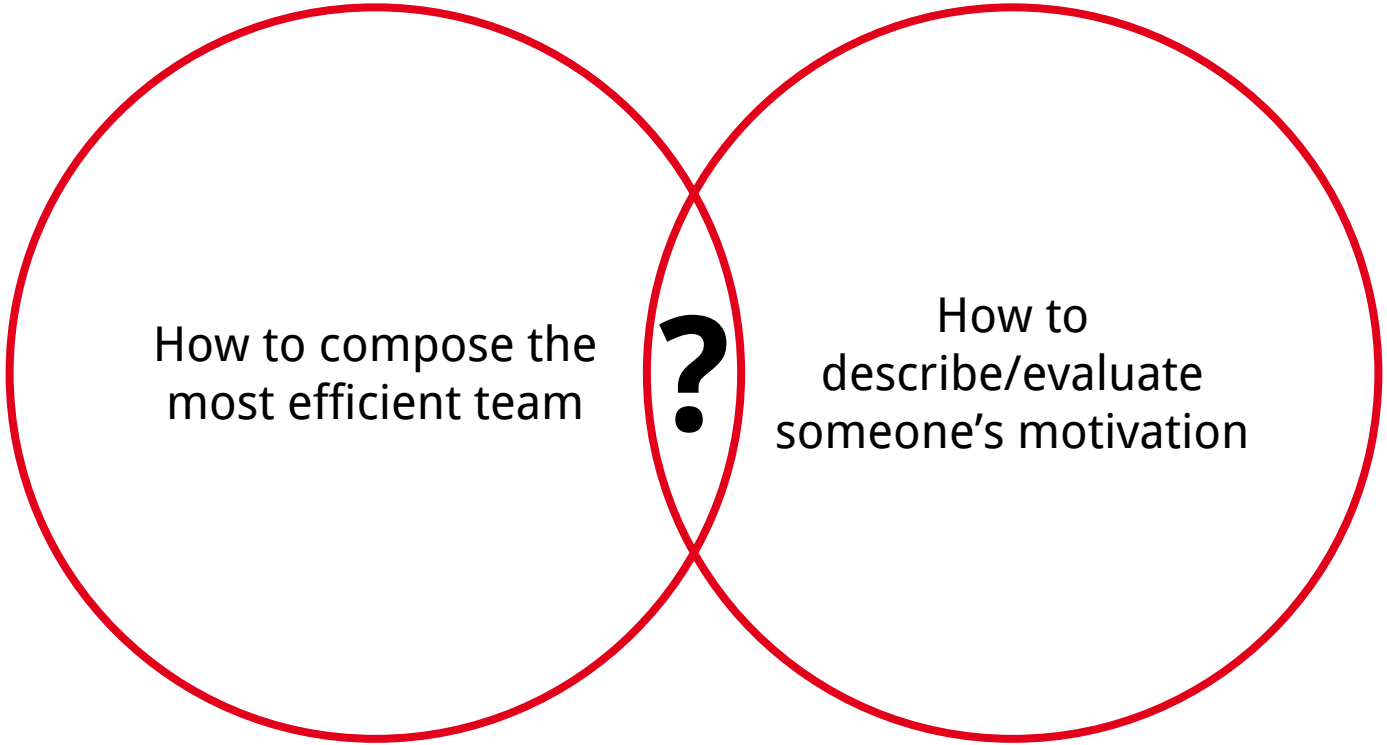
# Person Search (Outline)


- Sort the employees by their fitness scores
- Implicit recommendation
  - Best match → First position

# Fitness Score (Research)



# Fitness Score (Research)



# Fitness Score (Research)

- Ivanovska et. al: Algorithms for Effective Team Building
- Canós-Darós: An algorithm to identify the most motivated employees
  - General Motivation ↔ Task specific
  - Asking employees to rate their motivation generates suitable data
- Spoonamore et. al: Matching Sailors to Positions Based on Skill

# Matching Sailors to Positions Based on Skill

- Qualities
  - Fast to execute
  - Easy to implement and maintain
  - Understandable
- Factors included
  - Rating
  - Pay grade
  - NECs

# Matching Sailors to Positions Based on Skill

- Basic priciple: weighted mean of factors
  - $S = \alpha$ ratingscore + $\beta$ paygradescore + $\gamma$ NECscore
  - $\alpha, \beta, \gamma \geq 0$
  - $\alpha + \beta + \gamma = 1$

# Fitness Score (Basic Principle)

- Weighted mean of factors
  - Average skill level in searched skills
  - Average will level in searched skills
  - Specialization (skill levels)
  - Specialization (will levels)
- Estimation by users
  - Suitable data (Canós-Darós)
  - Focus on collaboration (external ratings → supervision)
- Weighting parameters configurable

# Fitness Score (Definitions and Helpers)

$$V = \{x \in \mathbb{N}_0^+ \mid 0 \leq x \leq 3\}$$

$$S = \{Java, Ruby, C++, ...\}$$

$$E = \{x \in S \mid \text{employee has skill x}\}$$

$$Q = \{x \in S \mid \text{user searches for skill x}\}$$

$$v_s : E \mapsto V$$

$$v_w : E \mapsto V$$

# Fitness Score (Factors)

- Skill levels in searched skills: $a_s$
- Will levels in searched skills: $a_w$
- Specialization in searched skills
  - Skill levels: $s_s$
  - Will levels: $s_w$

$$a_s = \left( \sum_{x \in E \cap Q} v_s(x) \right) \cdot \frac{1}{|E \cap Q|}$$

$$a_w = \left( \sum_{x \in E \cap Q} v_w(x) \right) \cdot \frac{1}{|E \cap Q|}$$

$$s_s = \frac{max(V) \ + a_s - \left( \left( \sum_{x \in E \setminus Q} v_s(x) \right) \cdot \frac{1}{|E \setminus Q|} \right)}{2max(V)}$$

$$s_w = \frac{max(V) \ + a_w - \left( \left( \sum_{x \in E \setminus Q} v_w(x) \right) \cdot \frac{1}{|E \setminus Q|} \right)}{2max(V)}$$

# Fitness Score (Basic Algorithm)

- Weighted mean of the factors
- Configurable weighting parameters ($w_{as}$, $w_{aw}$, $w_{ss}$ and $w_{sw}$)
  - Positive real numbers
  - Sum: 1
- Value between zero and one

$$f = \frac{w_{as} \cdot a_s}{max(V)} + \frac{w_{aw} \cdot a_w}{max(V)} + w_{ss} \cdot s_s + w_{sw} \cdot s_w$$

# Example

# Assumptions

- Weighting parameters
  - $w_{as} = w_{aw} = w_{ss} = w_{sw} = 0.25$
- Five persons; three known skills
- Distribution of skill/will levels not lifelike
- Notation: [skill level]/[will level]
- Searching for
  - Java and Ruby
  - In Hamburg

# Example (Java and Ruby in HH)


| Person | Location | Java | Ruby | C++ |
|---|---|---|---|---|
| Alice | Hamburg | 2/1 | 2/2 | 3/3 |
| Bob | Hamburg | 2/3 | 0/3 | 0/1 |
| Charlie | Hamburg | 3/3 | 2/1 | 1/2 |
| Donald | Hamburg | 3/3 | - | 2/2 |
| Erika | Frankfurt | 1/1 | 2/3 | 3/1 |

# Example (Java and Ruby in HH)


| Person | Location | Java | Ruby | C++ |
|---|---|---|---|---|
| Alice | Hamburg | 2/1 | 2/2 | 3/3 |
| Bob | Hamburg | 2/3 | 0/3 | 0/1 |
| Charlie | Hamburg | 3/3 | 2/1 | 1/2 |
| Donald | Hamburg | 3/3 | - | 2/2 |
| Erika | Frankfurt | 1/1 | 2/3 | 3/1 |

# Example (Java and Ruby in HH)


| Person | Location | Java | Ruby | C++ |
|---|---|---|---|---|
| Alice | Hamburg | 2/1 | 2/2 | 3/3 |
| Bob | Hamburg | 2/3 | 0/3 | 0/1 |
| Charlie | Hamburg | 3/3 | 2/1 | 1/2 |
| Donald | Hamburg | 3/3 | - | 2/2 |
| Erika | Frankfurt | 1/1 | 2/3 | 3/1 |

# Example (Java and Ruby in HH)


| Person | Location | Java | Ruby | C++ |
|---|---|---|---|---|
| Alice | Hamburg | 2/1 | 2/2 | 3/3 |
| Bob | Hamburg | 2/3 | 0/3 | 0/1 |
| Charlie | Hamburg | 3/3 | 2/1 | 1/2 |
| Donald | Hamburg | 3/3 | - | 2/2 |
| Erika | Frankfurt | 1/1 | 2/3 | 3/1 |

# Example (Java and Ruby in HH)


| Person | Location | Java | Ruby | C++ |
|---|---|---|---|---|
| Alice | Hamburg | 2/1 | 2/2 | 3/3 |
| Bob | Hamburg | 2/3 | 0/3 | 0/1 |
| Charlie | Hamburg | 3/3 | 2/1 | 1/2 |
| Donald | Hamburg | 3/3 | - | 2/2 |
| Erika | Frankfurt | 1/1 | 2/3 | 3/1 |

# Example (Java and Ruby in HH)


| Person | Location | Java | Ruby | C++ | f |
|---|---|---|---|---|---|
| Alice | Hamburg | 2/1 | 2/2 | 3/3 | 0.44 |
| Bob | Hamburg | 2/3 | 0/3 | 0/1 | 0.71 |
| Charlie | Hamburg | 3/3 | 2/1 | 1/2 | 0.69 |
| Donald | Hamburg | 3/3 | - | 2/2 | - |
| Erika | Frankfurt | 1/1 | 2/3 | 3/1 | 2/3 |

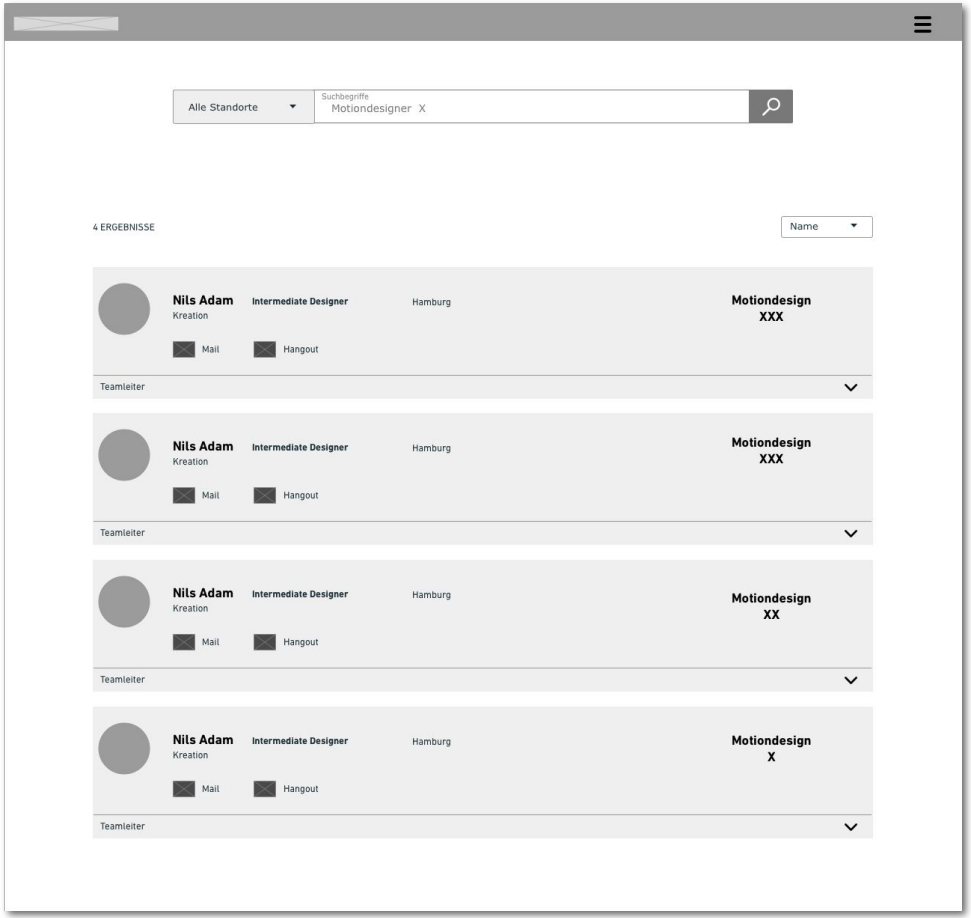
# Example (Java and Ruby in HH)

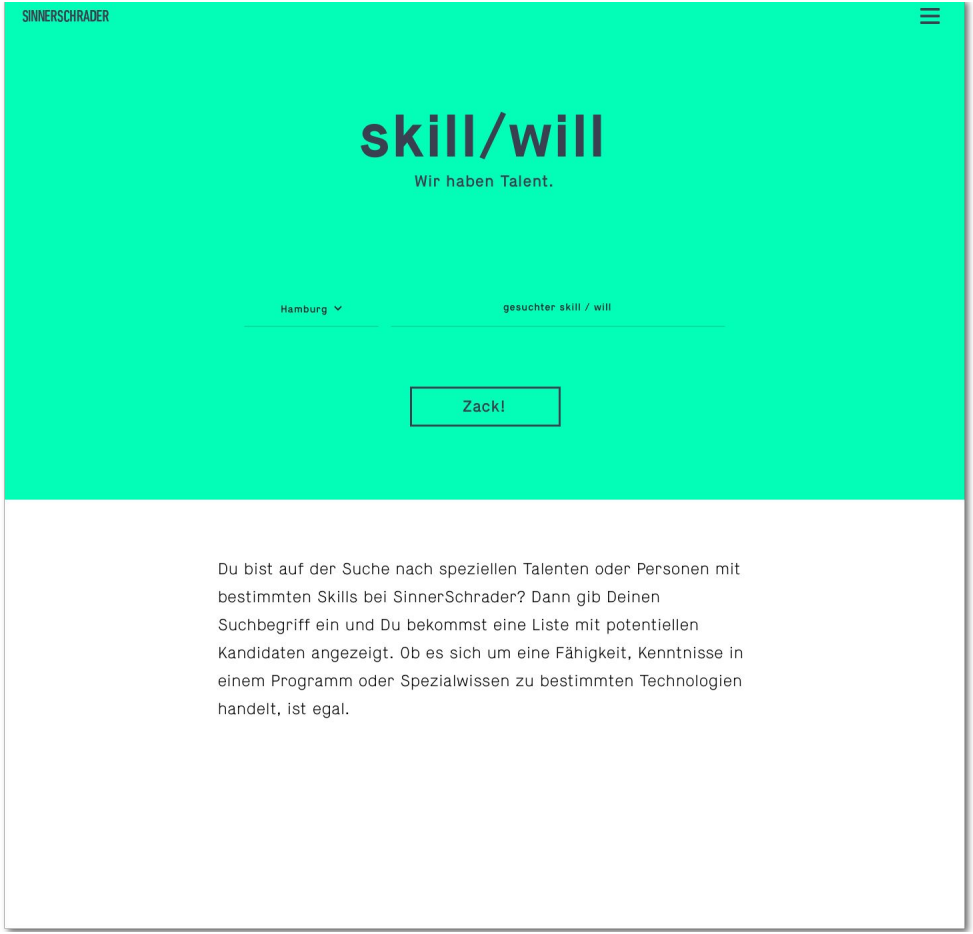

| # | Person | Location | Java | Ruby | C++ | f |
|---|---|---|---|---|---|---|
| 3 | Alice | Hamburg | 2/1 | 2/2 | 3/3 | 0.44 |
| 1 | Bob | Hamburg | 2/3 | 0/3 | 0/1 | 0.71 |
| 2 | Charlie | Hamburg | 3/3 | 2/1 | 1/2 | 0.69 |
| | Donald | Hamburg | 3/3 | - | 2/2 | - |
| | Erika | Frankfurt | 1/1 | 2/3 | 3/1 | 2/3 |

# Example (Java and Ruby in HH)


| # | Person | Location | Java | Ruby | C++ | f |
|---|---|---|---|---|---|---|
| 3 | Alice | Hamburg | 2/1 | 2/2 | 3/3 | 0.44 |
| 1 | Bob | Hamburg | 2/3 | 0/3 | 0/1 | 0.71 |
| 2 | Charlie | Hamburg | 3/3 | 2/1 | 1/2 | 0.69 |
| | Donald | Hamburg | 3/3 | - | 2/2 | - |
| | Erika | Frankfurt | 1/1 | 2/3 | 3/1 | 2/3 |

# Interlude:
# Visual Concept

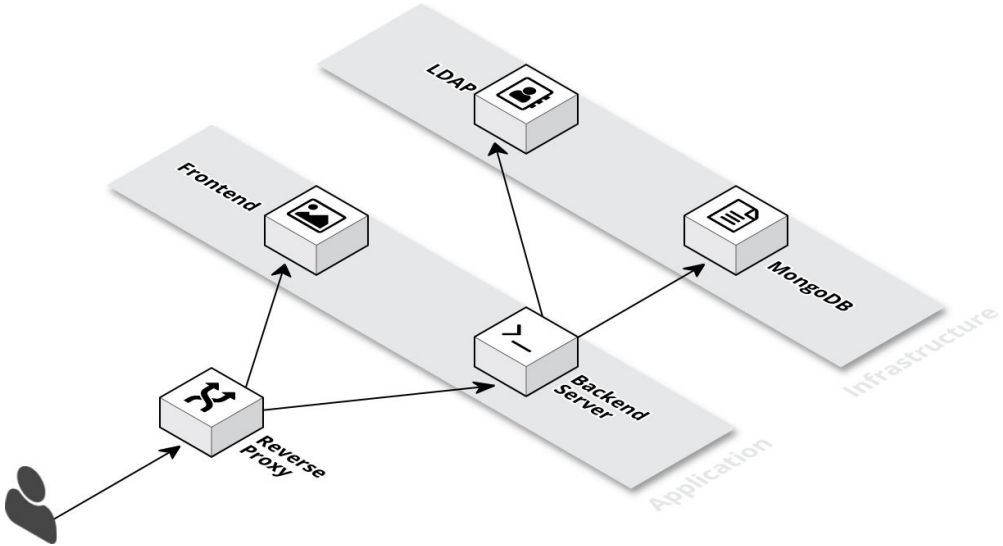

Alle Standorte
Suchbegriffe
Motiondesigner X
4 ERGEBNISSE
Name
Nils Adam
Intermediate Designer
Kreation
Hamburg
Motiondesign
XXX
Mail
Hangout
Teamleiter
Nils Adam
Intermediate Designer
Kreation
Hamburg
Motiondesign
XXX
Mail
Hangout
Teamleiter
Nils Adam
Intermediate Designer
Kreation
Hamburg
Motiondesign
XX
Mail
Hangout
Teamleiter
Nils Adam
Intermediate Designer
Kreation
Hamburg
Motiondesign
X
Mail
Hangout
Teamleiter


SINNERSCHRADER
skill/will
Wir haben Talent.
Hamburg
gesuchter skill / will
Zack!
Du bist auf der Suche nach speziellen Talenten oder Personen mit bestimmten Skills bei SinnerSchrader? Dann gib Deinen Suchbegriff ein und Du bekommst eine Liste mit potentiellen Kandidaten angezeigt. Ob es sich um eine Fähigkeit, Kenntnisse in einem Programm oder Spezialwissen zu bestimmten Technologien handelt, ist egal.

# Implementation

# Structure

- Reverse Proxy
  - Delivers frontend
  - Forwards to backend
- Backend Server
  - Retrieves,
  - Filters and
  - Modifiers data
- MongoDB
  - Skill data
  - Personal contact data (synced)
- LDAP
  - Personal contact data



# Backend Technologies

- Java
- Spring Boot
- Spring Data
  - MongoDB connection
- unboundid SDK
  - LDAP connection
- Swagger
  - Interactive API documentation
- JUnit
- Maven

# REST API

- Representational State Transfer
- JSON
- Access to all features of the backend

# REST API Endpoints (Login)

| URL | Method | Feature |
|---|---|---|
| /login | POST | User Login |
| /logout | POST | User Logout |

- Login to edit user's skills
- No login needed to search

# REST API Endpoints (Users)

| URL | Method | Feature |
|---|---|---|
| /users | GET | Main person search function |
| /users/{user} | GET | Get specific user's details |
| /users/{user}/skills | POST | Add/Update user's skills |
| /users/{user}/skills | DELETE | Remove skill from user's profile |

- New users added on their first login
- No removing of users

# REST API Endpoints (Skills)

| URL | Method | Feature |
|---|---|---|
| /skills | GET | Get all skills/text autocomplete |
| /skills | POST | Create new Skill |
| /skills/next | GET | Recommend next skill to enter |
| skills/{skill} | PUT | Edit skill (rename) |
| skills/{skill} | DELETE | Delete skill |

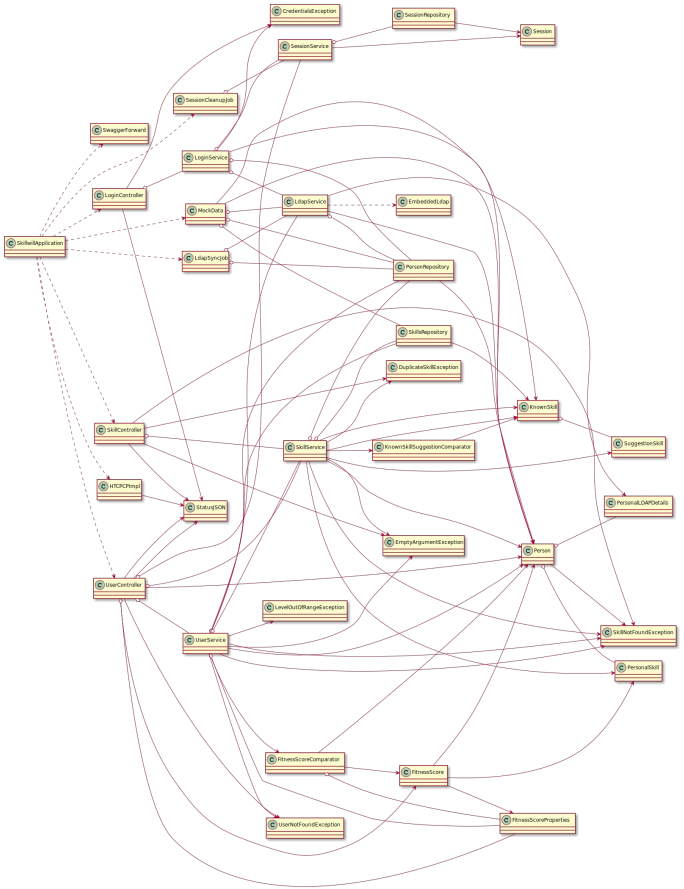
# REST API Endpoints (Skills)

| URL | Method | Feature |
|---|---|---|
| /skills | GET | Get all skills/text autocomplete |
| /skills | POST | Create new Skill |
| /skills/next | GET | Recommend next skill to enter |
| skills/{skill} | PUT | Edit skill (rename) |
| skills/{skill} | DELETE | Delete skill |

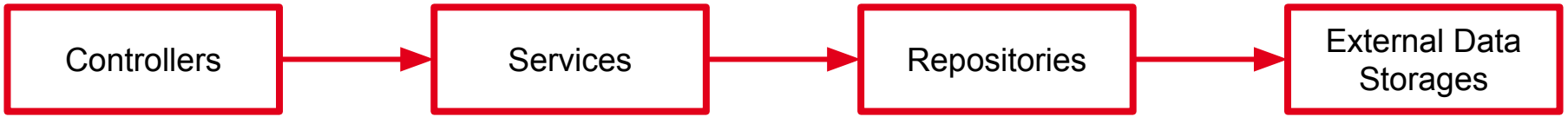
- Next skill to enter: Will be used in search view
- Separate Recommender System
  - Markov Chain
  - Item Based Filtering (Collaborative)

# Backend Structure

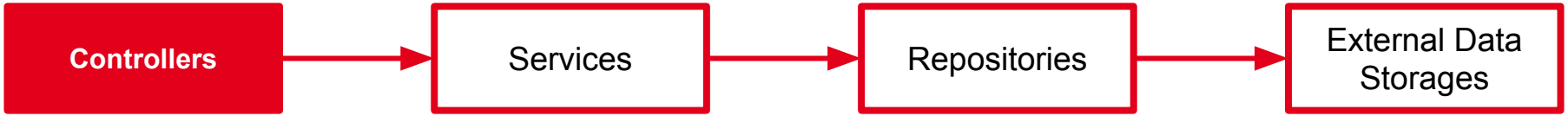
- Controllers
- Services
- Repositories
- Jobs
- Helpers
- Data Types



# Backend Structure

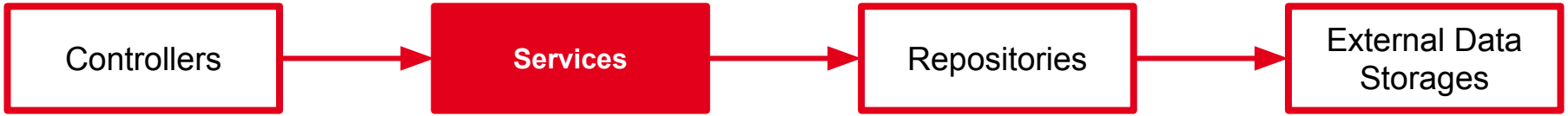


# Backend Structure



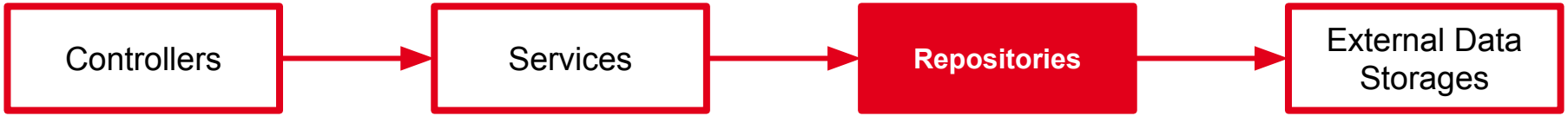

- Listen to API endpoints
- Use services to
  - Get data
  - Send data
  - Send commands
- Convert Objects to JSON

# Backend Structure



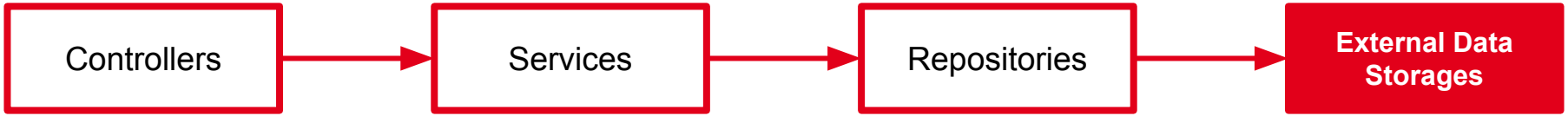

- Get, filter, transform, edit data
- Use repository objects to
  - Retrieve data from external sources
  - Write data to external sources
- Contain business logic

# Backend Structure




- Spring Data repository objects
- Provide Methods for CRUD operations
- Wrappers to simplify storage access

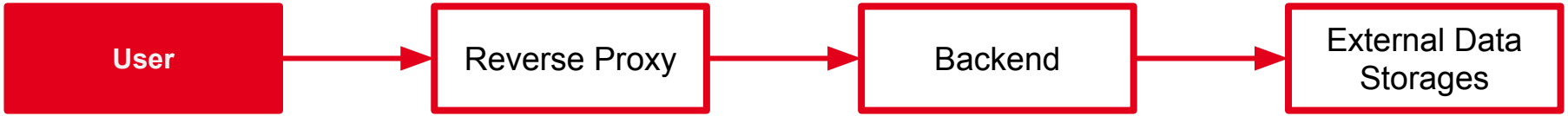
# Backend Structure




- Store Data
- MongoDB
  - Registered Skills (including transition lists for suggestion)
  - Persons
    - Personal Data (synced from LDAP)
    - Personal Skills (ID, skills with levels)

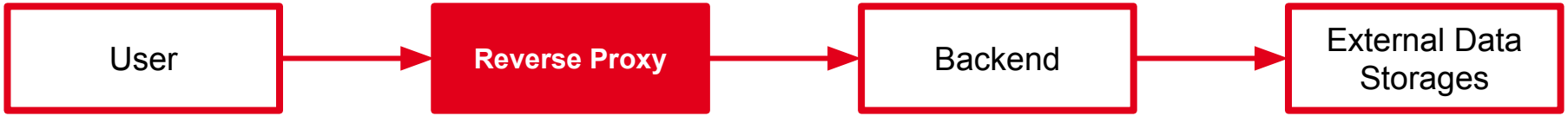
# Example

# API Call (Search for Java and Ruby in HH)



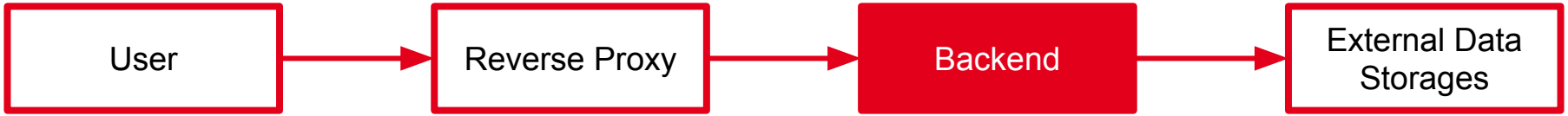

- Search for Java and Ruby in HH
- Browser Calls API
  - *api.some.tld/users?skills=Java,Ruby&location=Hamburg*

# API Call (Search for Java and Ruby in HH)



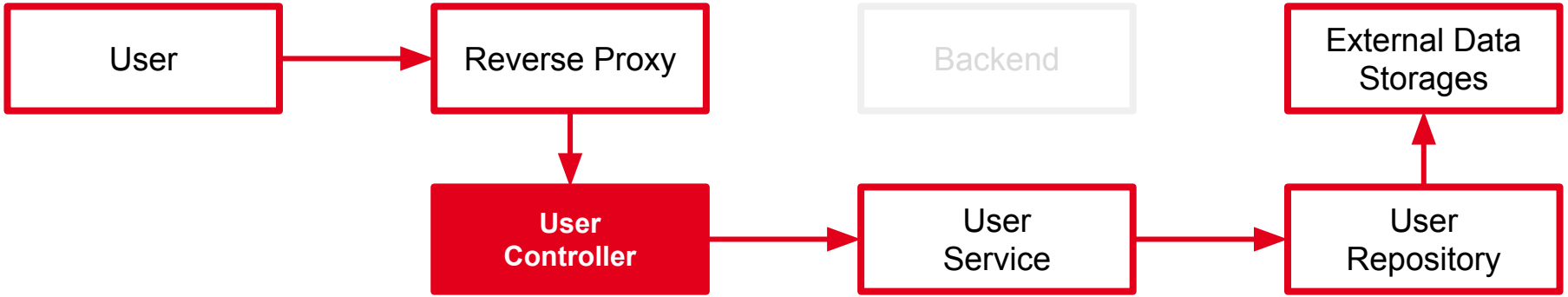

- Recognize API Call (Domain)
- Forward to Backend Server

# API Call (Search for Java and Ruby in HH)



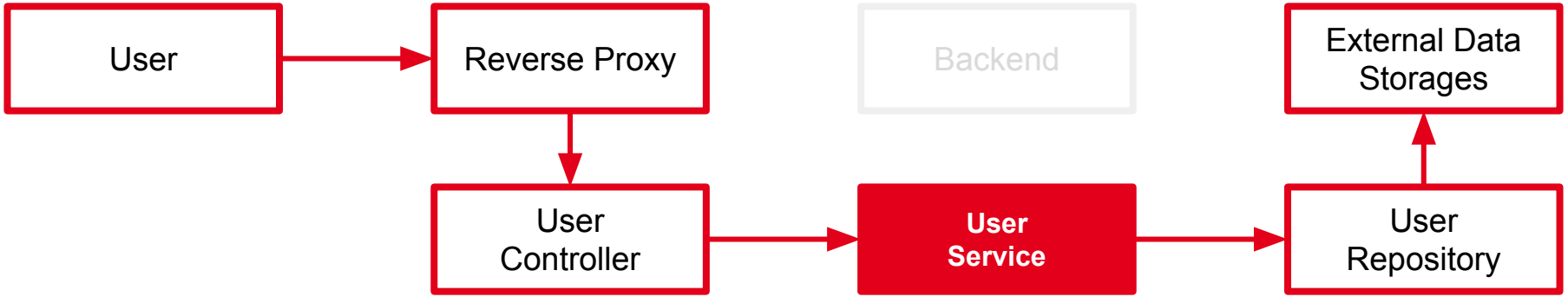

- Waits for HTTP Requests
- Dispatching to Controller

# API Call (Search for Java and Ruby in HH)



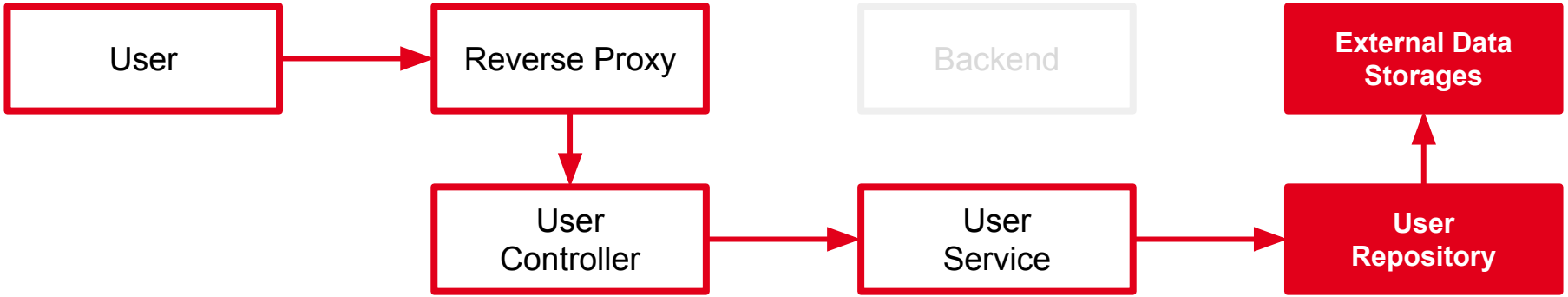

- Call method based on request URL and Parameters
  - /users/
  - ?skills=Java,Ruby&location=Hamburg
- Request matching persons from UserService

# API Call (Search for Java and Ruby in HH)



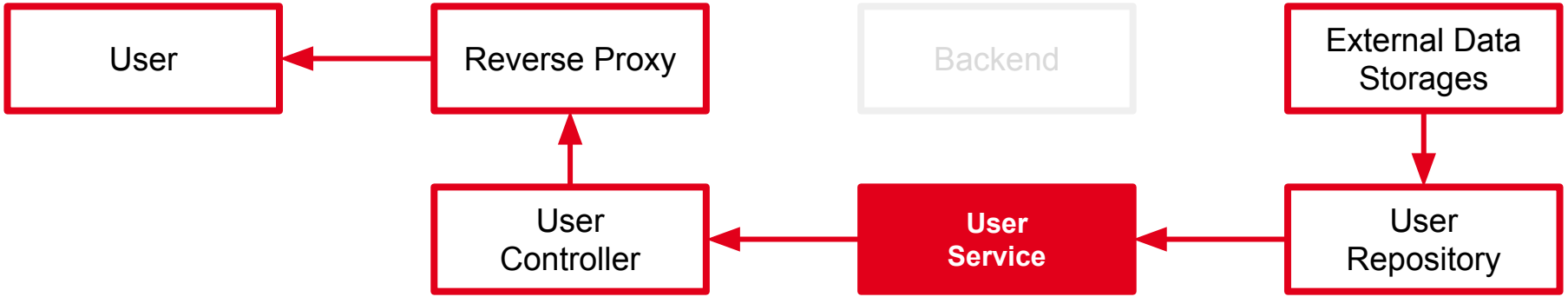

- Request needed users from Repository

# API Call (Search for Java and Ruby in HH)



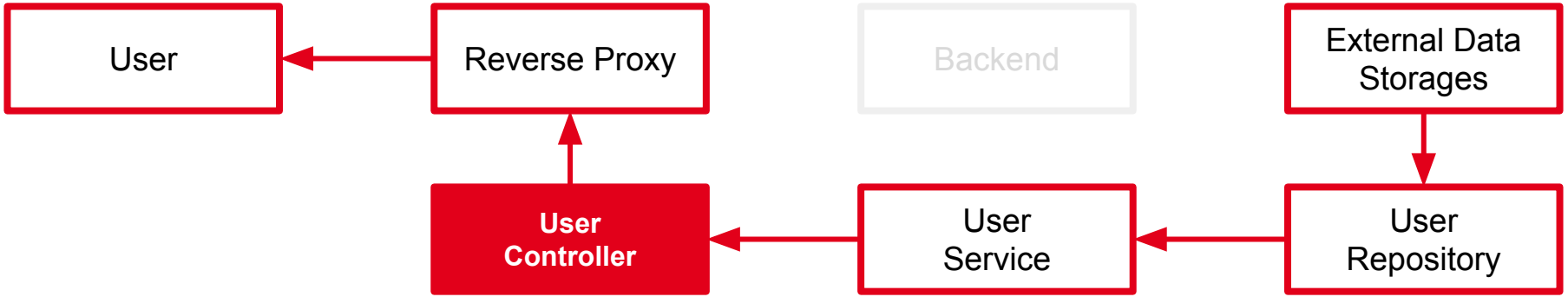

- Get data from MongoDB
- Return user objects to UserService

# API Call (Search for Java and Ruby in HH)



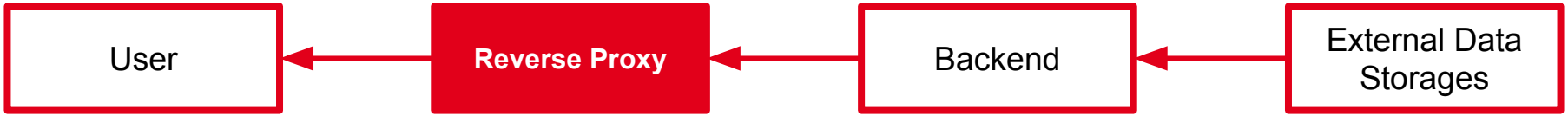

- Process retrieved user objects
- Apply search and fitness score algorithms

# API Call Cycle (Search for Java and Ruby in HH)



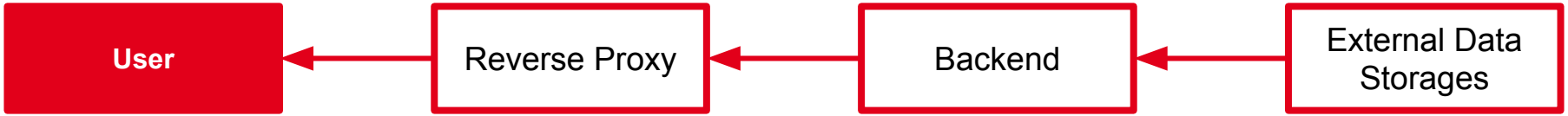

- Convert found user objects to JSON
- Return HTTP Response
  - In case of error, return corresponding HTTP Code

# API Call Cycle (Search for Java and Ruby in HH)




- Forward JSON response to client

# API Call Cycle (Search for Java and Ruby in HH)




- Parse JSON response
- Render result list

# Evaluation

# Recap: Requirements

- Functional
  - Person Search
  - Login, Logout
  - Users can modify their skills
  - Add, rename and delete registered skills
- Non functional
  - Device Support
  - Browser Support
  - Scalability
  - Response Times

# Recap: Requirements

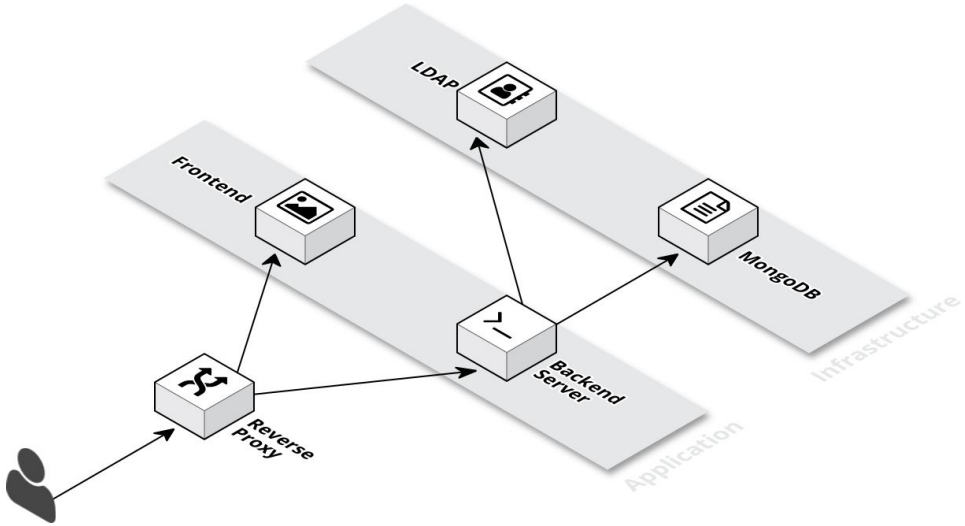
- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support
  - Browser Support
  - Scalability
  - Response Times

# Recap: Requirements

- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability
  - Response Times

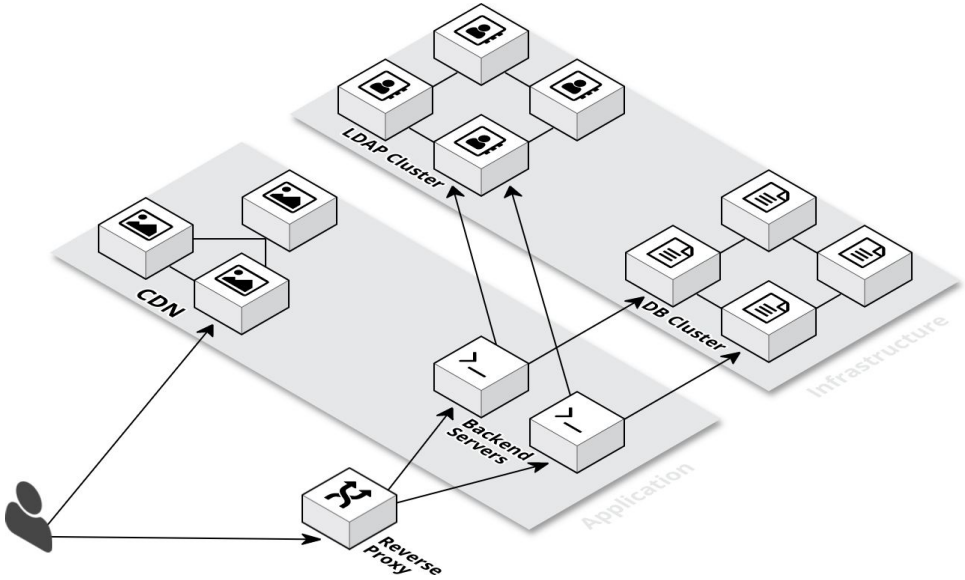
# Evaluation (Scalability)

- MongoDB
  - Designed and shown to be scalable
- LDAP
  - Six servers
  - Cluster is transparent to application
- Frontend
  - CDN
- Backend
  - Stateless Application
  - Reverse proxy as load balancer
  - Tested



# Evaluation (Scalability)

- MongoDB
  - Designed and shown to be scalable
- LDAP
  - Six servers
  - Cluster is transparent to application
- Frontend
  - CDN
- Backend
  - Stateless Application
  - Reverse proxy as load balancer
  - Tested



# Recap: Requirements

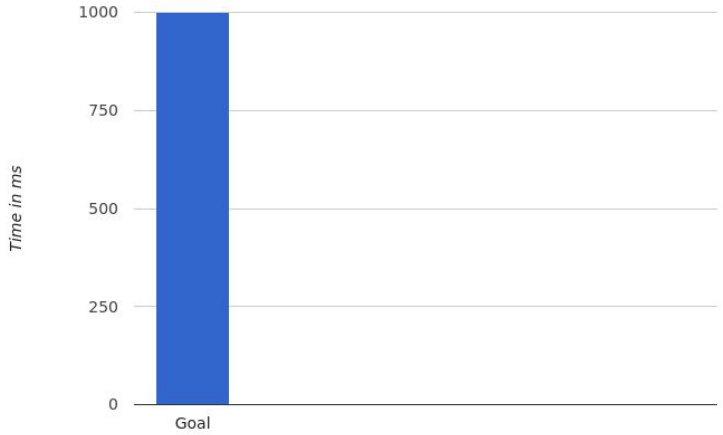
- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability
  - Response Times

# Recap: Requirements

- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability ✔
  - Response Times

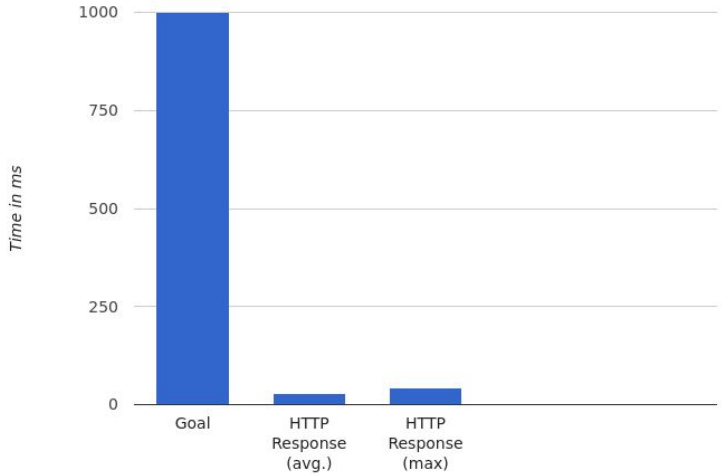
# Evaluation (Response Times)

- Goal: 1s max to finish rendering



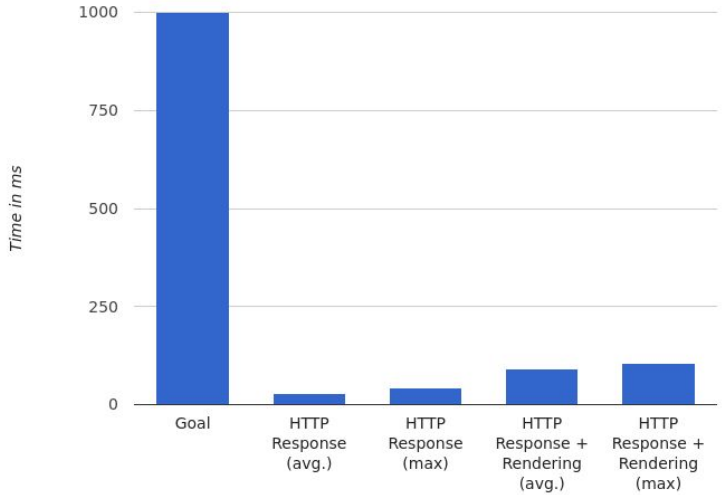
# Evaluation (Response Times)

- Goal: 1s max to finish rendering
- HTTP Response: 33ms/44ms (40 samples)



# Evaluation (Response Times)

- Goal: 1s max to finish rendering
- HTTP Response: 33ms/44ms (40 samples)
- + Rendering: 90ms/106ms (16 samples)



# Recap: Requirements

- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability ✔
  - Response Times

# Recap: Requirements

- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability ✔
  - Response Times ✔

# Evaluation (Fitness Score Algorithm)

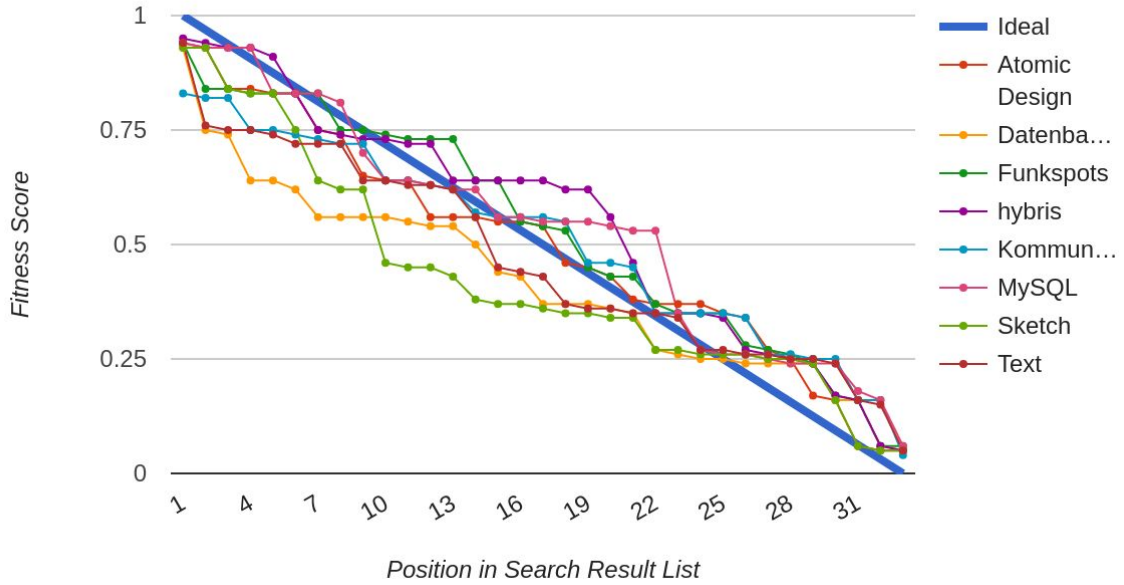
- Are fitness values distributed uniformly when the input values are?
  - No → the algorithm probably is not fair
  - Yes → we still do not know
- Is there a configuration of weighting parameters that reflects the users' perception?

# Evaluation (Distribution of Fitness Scores)

- Generate 100 users
- Assign a random number (n ≤ 17) of skills
- Assign random skill/will level for each skill
- Perform Searches
- Hypothesis to validate:
  - Random values are distributed uniformly
  - Fitness of employee's is distributed uniformly
  - Fitness Scores are distributed uniformly
  - Results sorted by fitness:
    - Linear falling function
    - f(first) = 1
    - f(last) = 0

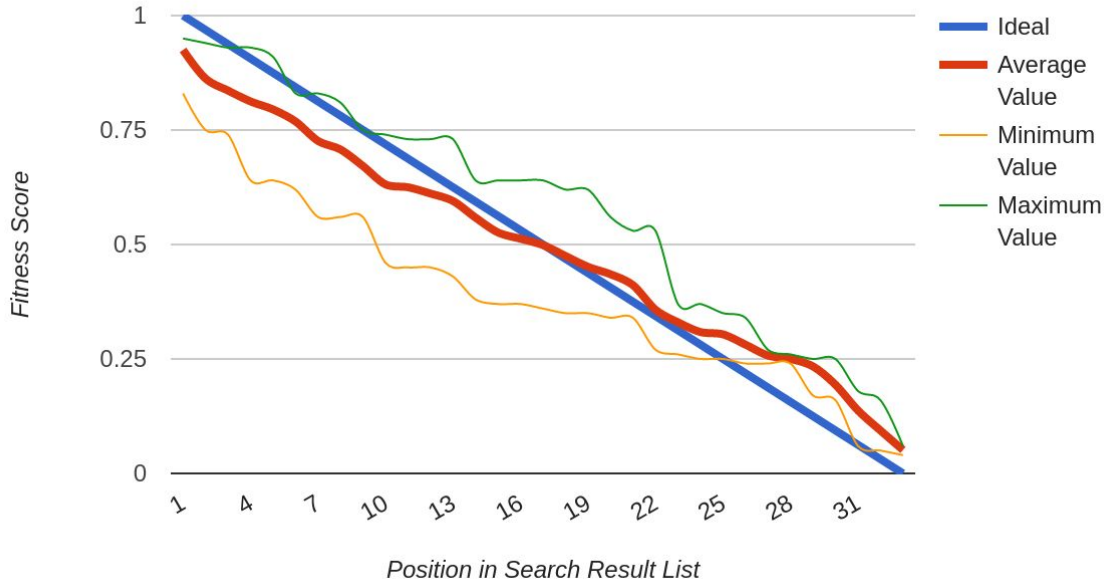
# Evaluation (Distribution of Fitness Scores)

- Searches for 8 distinct skills (33 results each)



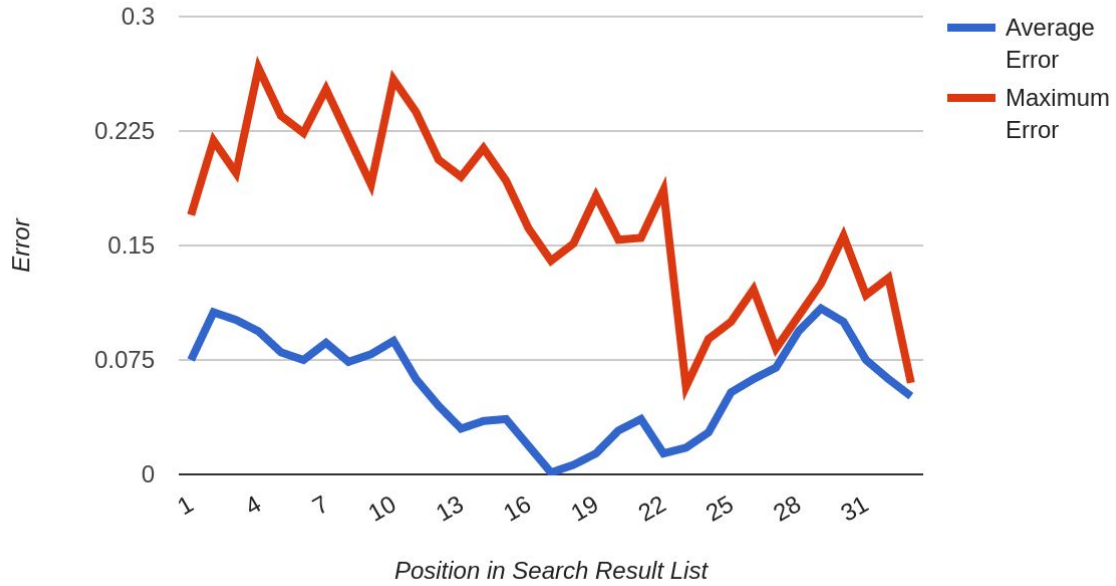
# Evaluation (Distribution of Fitness Scores)

- Searches for 8 distinct skills (33 results each)
- Average correlates with ideal



# Evaluation (Distribution of Fitness Scores)

- Average Deviation: 6%
- Maximum Deviation: 27%



# Evaluation (Distribution of Fitness Scores)

- Input distributed uniformly → Output distributed uniformly
- Sources of error
  - Random generation
  - Small sample (8 rows; 33 values each)

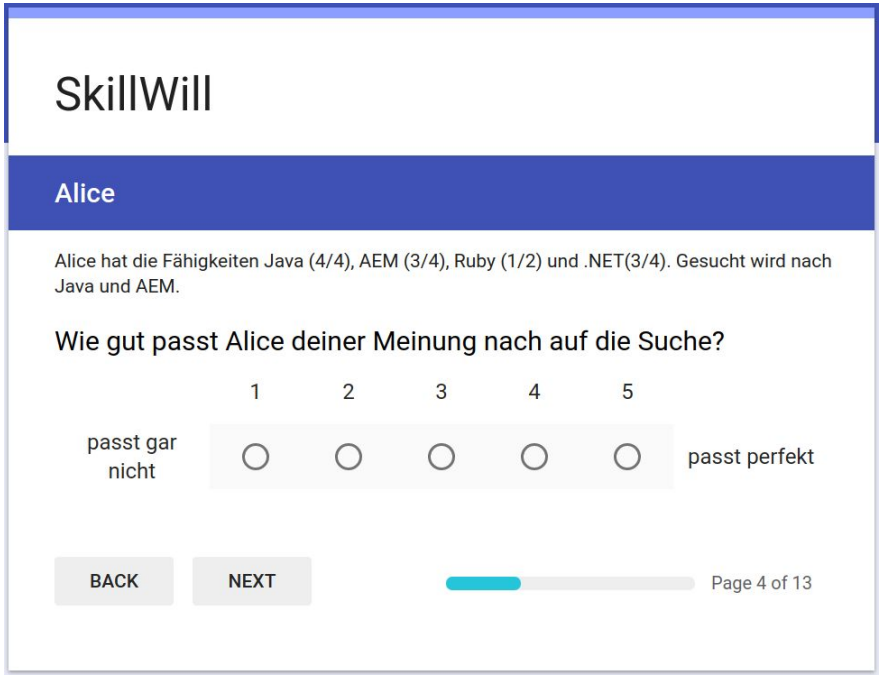
# Evaluation (Reflection of Users' Perception)

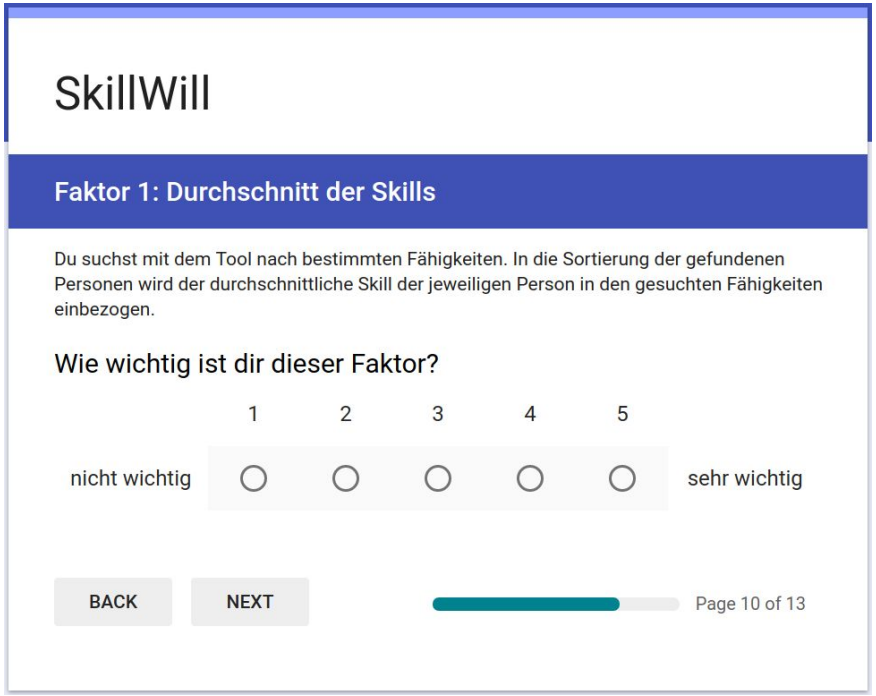
- Is there a configuration of weighting parameters that reflects the users' perception?
  - Employees estimate the importance of the factors
  - Employees estimate fictional persons' scores
  - Find a configuration that fits both

# Evaluation (Survey)

- Online Survey (Google Forms)
- Random group of 161 employees invited (35%)
- 41 participants in 72hrs (8%)
- Question Type 1
  - Fictional person, has skills
  - Specific Search
  - Subject estimates fitness  (Likert Items: 1 - 5)
- Question Type 2
  - Description of a factor included in the algorithm
  - Subject rates importance (Likert Items: 1 - 5)
- Type 2 shown after Type 1 → Knowledge about factors must not influence perception of fitness

# Evaluation (Survey)

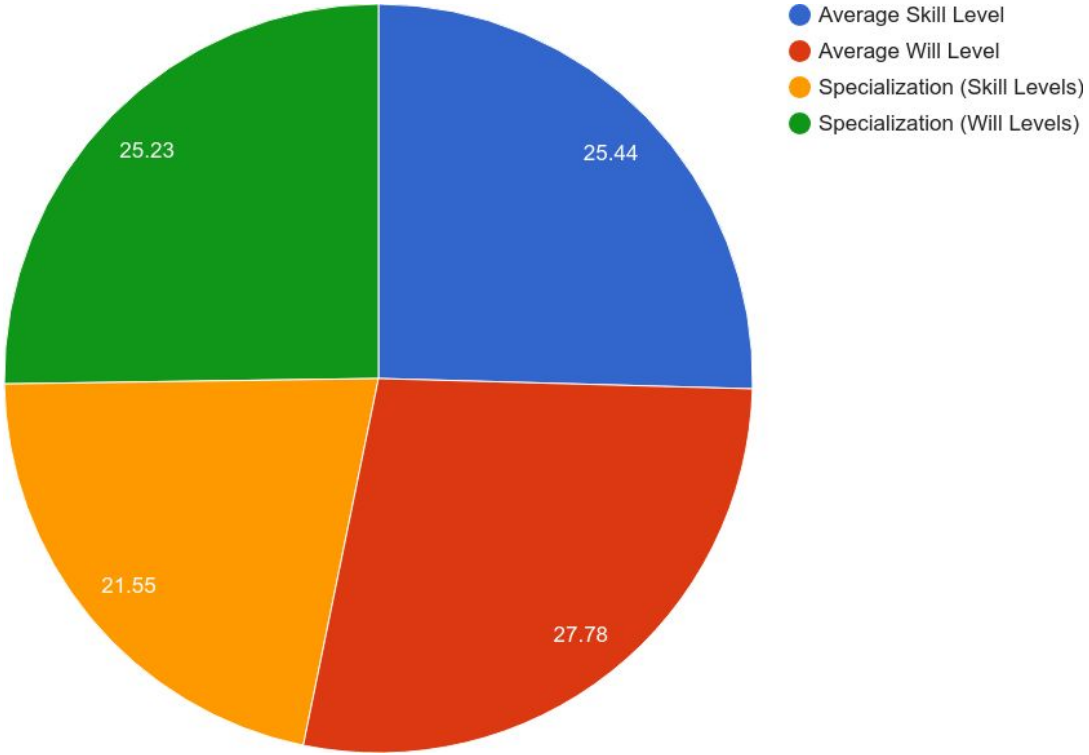


# Evaluation (Survey)

| Factor | Description | Mean Rating (1-5) | Mean Rating (%) |
|---|---|---|---|
| $w_{as}$ | Average skill level in searched items | 3.80 | 25.44 |
| $w_{aw}$ | Average will level in searched items | 4.15 | 27.78 |
| $w_{ss}$ | Specialization (Skill Levels) | 3.22 | 21.55 |
| $w_{sw}$ | Specialization (Will Levels) | 3.77 | 25.23 |

# Evaluation (Survey)

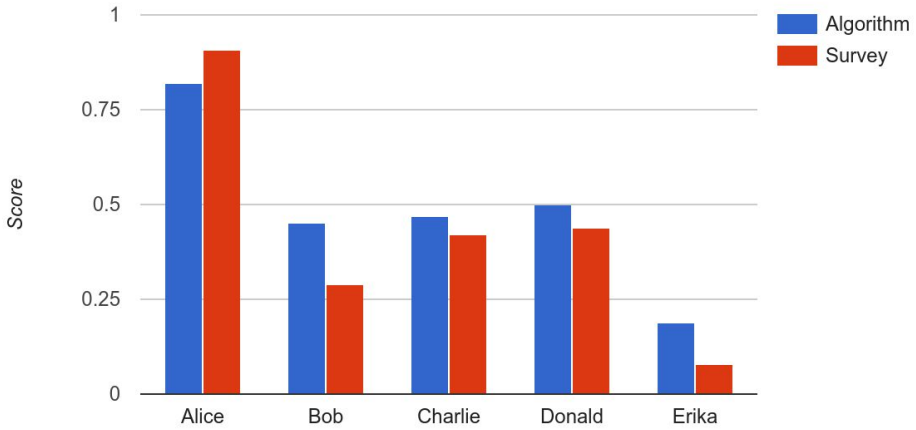


# Evaluation (Survey)

- Configure algorithm to use these weighting parameters
- Calculate fitness scores for the five fictional test persons
- Test, if there is a significant deviation
  - Two Tailed Heteroscedastic T-Test
  - Significance Level: $p \geq 0.1$

# Evaluation (Survey)

| Test Record | $f_a$ | $f_s$ | Dev. | $p \geq 0.1$ |
|---|---|---|---|---|
| Alice | 0.82 | 0.91 | 0.12 | No |
| Bob | 0.45 | 0.29 | 0.16 | No |
| Charlie | 0.47 | 0.42 | 0.16 | No |
| Donald | 0.50 | 0.44 | 0.17 | No |
| Erika | 0.19 | 0.08 | 0.18 | No |



# Evaluation (Survey)

- $p \geq 0.1 \rightarrow$ No significant deviation
- $p \geq 0.05 \rightarrow$ ⅖ Rows deviate significantly
- General algorithmic principle works

# Refining the Fitness Score Algorithm

- Estimated Factors ≅ 0.25
- Setting all to 0.25
  - Simplifies algorithm
  - No drastic effect on accuracy

$$f = \frac{w_{as} \cdot a_s}{max(V)} + \frac{w_{aw} \cdot a_w}{max(V)} + w_{ss} \cdot s_s + w_{sw} \cdot s_w$$

$$\Rightarrow f = \frac{a_s + a_w}{4max(V)} + \frac{s_s + s_w}{4}$$

# Recap: Requirements

- Functional
  - Person Search
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability ✔
  - Response Times ✔

# Recap: Requirements

- Functional
  - Person Search ✔
  - Login, Logout ✔
  - Users can modify their skills ✔
  - Add, rename and delete registered skills ✔
- Non functional
  - Device Support ?
  - Browser Support ?
  - Scalability ✔
  - Response Times ✔

# Demo

# Demo (Fallback)

# Résumé & Outlook

# Résumé

- Context: SinnerSchrader
- Requirements
  - Collaboration ↔ Supervision
  - Focus: search
  - Includes motivation
- Commercial solutions
  - Do not fulfill requirements
- Concept
  - Search and fitness score algorithms
- Implementation
- Evaluation
  - Basic ideas behind algorithms work
  - Could be tweaked to get even better results

# Outlook

- Tweaking of  algorithms
- Last touches on the frontend
- Test phase with real users
- Legal concerns
- Extend with features
  - Skill categories
  - Search in departments
  - ...

# Thanks

# Thanks
# for all the fish

# Find this on Github!

github.com/t0rbn/BSc
github.com/sinnerschrader*

*soon

# Q&A