DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Reciprocal Recommender System for Joint Event Visits

Michael Wang

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Reciprocal Recommender System for Joint Event Visits

Ein wechselseitiges Empfehlungssystem für gemeinsame Veranstaltungsbesuche

Author:	Michael Wang
Supervisor:	Prof. Dr.-Ing. Jörg Ott
Advisor:	M.Sc. Daniel Herzog
Submission Date:	July 15th, 2019

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, July 15th, 2019 Michael Wang

Abstract

Event recommenders present users with events they find interesting. These events can become more interesting for users depending on who will join them in the visit of the event. Thus, social matching features could improve the accuracy of event recommenders. This thesis investigates whether this prospect is true.

We extended a pre-existing event recommender ("*Eventguide*") with social matching features. For this purpose, we enhanced the recommender's algorithm with three stages: first, by considering demographic information; second, by finding similar users; third, by adapting a reciprocal algorithm that computes reciprocal matches ("*RECON*").

Our evaluation with 20 participants showed that they did not find social matching to be significantly beneficial. The main reason for this is that participants were not actively interested in getting to know other people. Thus, our study raised further research questions, for example, whether people who are actively interested in getting to know other people find social matching features in event recommenders interesting. If so, one should also research what the ratio of these people to the total potential user base is.

Contents

1. Introduction

This chapter introduces the motivation for this thesis and describes the problem statement. We also state the research questions, then explain our approach to answering these questions, and finally, present a thesis outline.

1.1. Motivation

Reciprocal recommender systems recommend people to people [49]. They can be used in various domains, such as for dating, employment, or mentoring recommendations.

Another domain of interest is event recommendation. Various contextual criteria influence the attractiveness of an event, such as its content, location, and timing [36]. Another criterion is sociality, i.e. with whom a person will go to an event with. For some people this may indeed be the most important factor in whether one will participate.

People like to visit events together, and socializing is a prominent factor for why people attend events [45]. Indeed some events are heavily dependent on such social interactions, for example, in amateur team sports or escape room events. Event recommenders can thus be improved by not only recommending events, but by also matching users with people who would like to join them in visiting the event.

1.2. Problem Statement

In this thesis, we extend an event recommender to include social matching functions. For this purpose, we enhance a pre-existing event recommender called "*Eventguide*" which is a mobile app previously implemented by Herzog [29] and then developed further by our industry partner 4A Solutions [1]. This app recommends events in the area of Munich, Germany; and it employs a "content-boosted collaborative filtering" [30] algorithm. Using the *Eventguide* provides us with some advantages:

First, Munich has a high ratio of single households at 55% in the year 2016 [61], compared to 41% in Germany and 33% in the European Union [26]. This leads us to the assumption that Munich people are a good target group for socialization features.

Second, the event recommender already has an existing user base and we aim to increase the number of users by improving the system with social matching features.

1.3. Research Questions

We integrate social matching functions into the *Eventguide* because we want to research to which extent these features provide value. The following research questions (RQ) look at this goal from different perspectives:

- **RQ1:** Are users more interested in events with social matching?
- **RQ2:** Do recommendations improve with social matching?
- **RQ3:** Do social matching features negatively impact the usability?
- **RQ4:** What influences do demographic factors have?

These research questions will be explained in detail in Chapter 6.

1.4. Approach

To answer these research questions, we approach the problem in three steps:

First, we elicitate requirements to enhance the *Eventguide* with social matching functions. For this purpose, we analyze the app's current user base and examine existing approaches. We can then define functional requirements in the form of user stories and create wireframes that model the user interface.

Second, we implement a prototype of this system according to the user stories and the wireframes. For this purpose, we extend the pre-existing event recommender app, enhance the server, and integrate reciprocal social matching algorithms.

Third, this prototype is evaluated in a field study. We explain how we derived the research questions and then answer them, detailing the study objects, procedure, and results. Finally, we discuss the results.

1.5. Thesis Outline

This thesis evaluates to which extent social matching features are of value for event recommender systems. In Chapters 1 and 2, we explain why this is of interest and explain the necessary background knowledge. Chapter 3 then presents related work. Afterwards, in Chapter 4, we elicitate requirements for the implementation of a prototype, described in Chapter 5. This prototype is then evaluated in Chapter 6, in a field study. Finally, in Chapters 7 and 8, we present future work and summarize the contributions of this thesis.

2. Fundamentals

This chapter introduces fundamental knowledge needed for subsequent chapters by explaining the individual components of "a reciprocal recommender system for joint event visits": We first explain recommender systems, then detail event recommenders, and finally present why reciprocal recommenders are special.

2.1. Recommender Systems

Recommender systems are "software tools and techniques that provide suggestions for items that are most likely of interest to a particular user" [55]. Manifold items can be recommended by such systems, including products on e-commerce websites and movies on media streaming services. Resnick et al. explain that recommender systems can help users to "make choices without sufficient personal experience of the alternatives" [54]. They further go on to detail that users then do not need to rely on other people or the word of mouth.

One of the simplest ways to present personalized recommendations is to display the items (such as products or movies) in a ranked list [55]. This list can be generated in many ways; Burke, for example, distinguishes five different approaches [16]:

Content-Based

In content-based recommenders, users' preferences for items are compared to the database of items that can be recommended, assuming that similar items are also rated similarly [3]. To compute this, items are classified into characteristics: movies, for example, could have a director, a genre, and actors.

Before recommendations can be provided, users must first interact with the system and rate items so that the recommender can gather the user's preferences. Only then can the system recommend items based on the preferences. If a user, for example, liked horror movies, then the recommender could present similar horror movies. However, if the user is new to the system and has no ratings for items, then the system can only provide recommendations of limited quality. This problem of handling new users is known as the *cold-start* problem [42].

Another problem is the *stability vs. plasticity* problem: Burke explains that "[a] steak-eater who becomes a vegetarian will continue to get steakhouse recommendations [...], until newer ratings have the chance to tip the scales" [16].

Collaborative

In collaborative systems, recommendations for items are generated from other users' preferences [23]. These systems first find users that are similar to the active user ("neighbors"), and then recommend items that these neighbors have liked in the past [55]; this is known as a "neighborhood-based" approach.

It is also possible to use other algorithms: The "item-based" approach, for example, predicts an item's rating based on the ratings a user has given for similar items [58]. Ricci et al. describe that "[when] the number of users exceeds the number of available items, item-based approaches should be preferred since they provide more accurate recommendations [...]. On the other hand, user-based methods usually provide more original recommendations [...]." [55]

Similar to content-based approaches, collaborative systems also suffer from the cold-start problem, in fact they expand the problem of handling new users badly to also handling new items badly. Furthermore, they also suffer from the stability vs. plasticity problem. However, one advantage to content-based approaches is that items do not necessarily need to be classified by their characteristics.

Demographic

Demographic systems generate recommendations from a user's demographic profile, assuming that different demographic niches should receive different recommendations [55]. An e-commerce website could, for example, provide different products to users from different countries, or tailor recommendations to the users' age and gender.

Knowledge-Based

Knowledge-based recommenders suggest items "based on specific domain knowledge about how certain item features meet users' needs and preferences" [55], using case-based algorithms. They tend to perform better in the beginning, but may be surpassed by other methods, such as collaborative systems, in the long run [55].

Hybrid

Hybrid approaches combine two or more of the techniques described above to improve the deficiencies of one of the involved techniques (usually the cold-start problem) [16].

2.2. Event Recommenders

Event recommenders face additional challenges compared to systems that recommend other items: Minkov et al. explain that events are only valid for a short amount of time [42]. They go on to describe that "[b]y the time one can expect user feedback on a specific event, that event is no longer relevant: an event recommendation system therefore has to recommend items for which no explicit feedback exists". Minkov et al. also describe that events can be very diverse and that it is difficult to decide whether an event is similar to previous events. In conclusion, this leads to the problem that future events are difficult to recommend by traditional methods, so that additional data modelling techniques need to be performed: one could, for example, "pool together [the] feedback from users with similar preferences" [42].

Advanced event recommenders also consider the context to become *context-aware* systems: If the weather forecast was good, then outdoor events could be more appropriate, for example. Macedo et al. explain that contextual signals can overcome the cold-start problem in event recommenders, using e.g. location signals and temporal signals [36].

2.3. Reciprocal Recommenders

Reciprocal recommenders differ from other recommenders in that they do not recommend items to people, but people to people [49]. They can be used in various domains, such as for dating, employment, or mentoring recommendations. Pizzato et al. explain that successful recommendations only occur when both people like each other, i.e. when they reciprocate [49], and that reciprocal recommenders must thus consider the preferences of both users. Koprinska and Yacef describe the main differences between reciprocal recommenders and more traditional recommenders in Table 2.2 [32]:

Typical recommender	**Reciprocal recommender**
Success is determined solely by the user seeking the recommendation.	Success is determined by both users – the subject and object of the recommendation.
Users have no reason to provide detailed explicit user profiles.	Users are expected to provide detailed self-profiles. Explicit profiles and preferences are often inaccurate.
Satisfied users are likely to return for more recommendations. Better recommendations mean more engagement.	Users may leave the system after a successful recommendation. Better recommendations might mean less engagement.
The same item can be recommended to all users.	Popular users should not be recommended to too many users.

Table 2.2.: *Main differences between reciprocal and traditional recommenders, taken from [32]*

3. Related Work

This chapter describes related work, beginning with previous work of Herzog and Wörndl that we extend in this thesis. We also adapt work from Pizzato et al., this is also described. Afterwards, we present related work of Xia et al., Macedo et al., De Pessemier et al., and Liu et al.

Herzog and Wörndl

Herzog and Wörndl implemented an event recommender that is aware of the context and that uses a hybrid algorithm (content-boosted collaborative filtering) to generate recommendations for events [30].

Their paper is based on previous work by Herzog [29]. In his master's thesis, Herzog first conducted interviews with experts from the event industry to analyze the needs of the industry from different stakeholder perspectives, including event promoters, ticketing platforms, artists, and visitors. He then elicitated requirements for an event recommender system and designed paper prototypes for a mobile application, paying attention to the user experience by designing an intuitive interface that can be swiped through. Herzog also placed special value on the design of the recommender algorithm: for this, he adapted the hybrid content-boosted collaborative filtering algorithm of Melville et al. [41]. He also extended the system for context-awareness, including contexts such as the user's location, times of availability, or interest in genres.

Herzog then evaluated his implementation in a field study with 16 participants over a duration of 2 weeks. After two weeks, study participants were surveyed and most participants were satisfied with the recommender system (μ: 3.75, σ2: 0.83). He concludes that the hybrid algorithm of Melville et al. can be adapted for event recommendations in a promising way. He also finds that this algorithm can be extended by context-aware features. Finally, he proposed future work, including more contextual factors to be included in the hybrid algorithm, or social recommendations using social networking services.

This work is relevant, because we extend their event recommender system with social matching features. They have described how they implemented their system and we need to integrate into that system, including the pre-existing app and server.

Pizzato et al.

Pizzato et al. introduced RECON, a reciprocal recommender for online dating [49]. Their work was the first to recognize the special properties of reciprocal recommenders, some of which we already detailed in Table 2.2, e.g. that the success is determined by both users – the subject and object of the recommendation.

In their work, Pizzato et al. implement a content-based approach to calculate reciprocal matches: first, they find the user's preferences; second, they compute compatibility scores to other users; and finally, they create a list of reciprocal recommendations.

They evaluated their recommender with the database of an Australian dating website over a six week period. The data of the first four weeks are used for training, consisting of 1.4 million messages sent by 90,000 users; the remaining two weeks are for testing. Their results demonstrate the need of accounting for reciprocity: RECON improves the success rate of recommendations and mitigates the cold-start problem.

Their work is relevant, because we also implement a reciprocal recommender. We adapt their approach and their algorithm, and apply these to the domain of recommending events instead of online dating.

Xia et al.

Xia et al. improved upon RECON by introducing similarity measures that "characterize the attractiveness and interest between two users" [64]. They adapted RECON's algorithm and also computed reciprocal scores to measure the compatibility between users. Their evaluation sampled 200.000 users from a Chinese dating website and they show that "[their] recommendation algorithms significantly outperform previously proposed approaches, and [that] collaborative filtering-based algorithms achieve much better performance than content-based algorithms in both precision and recall" [64].

Their work is relevant, since their improvement on RECON shows how our algorithm (an adaptation of RECON) could also be improved.

Macedo et al.

Macedo et al. implemented a context-aware event recommender for user-created events in event-based social networks (EBSN) [36]. To overcome the cold-start problem, they exploited contextual signals available from EBSNs, including social, geographic, and temporal signals. Their evaluation in a crawl of Meetup.com then shows that multiple contexts improve the accuracy of recommendations and mitigate the cold-start problem.

Their work is relevant, since we both recommend events for social matching. However, we use a different approach: our recommender is reciprocal, thus we also need to consider the preferences of the users attending the events (unlike Macedo et al.).

De Pessemier et al.

De Pessemier et al. implemented a recommender that integrates a user's social network as a knowledge source [22]. For this purpose, their recommender selects which of the user's friends on Facebook would likely join him in the visit of the event.

Their work is relevant, since we both pursue the same goal, i.e. to enhance event recommendations by including other people that would join a user in the event visit. However, our our approach is different: we allow feature reciprocal matching of strangers, instead of uni-directional recommendations of Facebook-friends.

Liu et al.

Liu et al. introduced event-based social networks, i.e. social networks where users can "create, distribute, and organize social events" [34]. They found that event-based social networks include offline social interactions which are not present in more traditional social networks.

Their work is relevant, because they identified another way of combining events with social matching features. However, this thesis focuses on the improvement of event recommenders and not on social networks.

4. Requirements

In this chapter we elicit functional requirements for the implementation of a social matching and event recommendation system. We first explain the purpose of gathering requirements and present our methodology. Afterwards, we analyze the current user base of the *Eventguide*-App and examine existing approaches to social matching and event recommendation. These analyses help us in deriving functional requirements in the form of user stories. Finally, we create wireframes of the user interface and summarize the findings of this chapter.

4.1. Purpose and Methodology

In this section we first explain the purpose of gathering requirements and then present the methodology used in this elicitation process.

Applied to the context of recommender software systems, Robertson and Robertson [56] state that "requirements are what a software product [...] is meant to do and to be". They further add that requirements exist whether they are discovered or not, and that a product will never be right unless it conforms to its requirements. However, they also explain that the requirements elicitation process is not focusing on creating a requirements document. Instead, it focuses on discovering and understanding the real problem so that alternative solutions can be explored. They conclude that, "in essence, then, requirements are not about the written requirements as such, but rather an uncovering of the problem to be solved" [56].

To fully uncover the core problem, we gathered experts from the event industry, including the publisher of an event magazine and developers of event recommenders. This gathering kicked-off the project, defined the scope of the problem, explored and defined functional requirements, and gained a consensus on the goal of the project.

In preparation for this project kick-off meeting, we analyzed the current user base and created personas, i.e. fictional people representing parts of the user base, further detailed in Section 4.2. We also analyzed existing approaches within our problem domain, presented in Section 4.3. With these background of these two analyses, functional requirements were explored in the kick-off meeting. We refined these exploratory requirements further and defined user stories, presented in Section 4.4. Finally, we created wireframes of the user interface, showcased in Section 4.5.

4.2. User Analysis

This section analyzes the current user base of the *Eventguide*-App. Our implementation will extend this app; therefore the current users are also likely to be the first users of our implementation. With this analysis we gain insight into the personalities of the people that will be matched to each other. This data will be used in the kick-off meeting so that we can elicit better requirements. We first present and interpret the data used in this analysis. Afterwards, personas are derived to better discuss the backgrounds and motivations of our user base with the use of fictional characters.

4.2.1. Data

In order to analyze our user base, the *Eventguide*-App integrates the *Google Analytics for Firebase* framework which collects real user data, such as age, gender, and interests. These demographics data and interests data is collected through the *Android Advertising ID* and *iOS Identifier for Advertisers*, so if these identifiers are not present for a user, then that user will be missing from the data set [28]. *Firebase* was able to collect the data of 2,427 users in a span of 28 days, and Table 4.1a presents their demographics. Table 4.1b then describes the common interests of these users.

	Male	**Female**
18-24	4.5%	4.0%
25-34	16.0%	11.0%
35-44	17.5%	8.5%
45-54	15.5%	6.0%
55-64	8.0%	2.0%
65+	4.5%	2.5%
Sum	66.0%	34.0%

(a) Demographics

% of users	**Interests**
46.4%	Mobile Enthusiasts
44.2%	Shoppers
37.7%	Avid News Readers
34.4%	Avid Political News Readers
33.8%	Travel Buffs
32.9%	Green Living Enthusiasts
32.0%	Opera & Theater Aficionados
30.2%	Avid Investors

(b) Common interests

Table 4.1.: *Analytics data of 2,427 users*

4.2.2. Personas

From this data, we derive fictional people called personas that represent a majority of our initial user base. First, Table 4.1a shows that two-thirds of users are male and that the age of most users ranges from 25 to 54 years old. Furthermore, Table 4.1b suggests that users have money to spend, as they are interested in shopping, traveling,

and investing. Users also seem to be educated, as they are interested in news, politics, and green living. Lastly, many people are opera and theater aficionados. This is logical, because the *Eventguide*-App also suggests opera and theater events.

Three personas are modelled from the real data, as described in Table 4.3. The user base is two-thirds male; therefore we create two male personas and one female persona. Different characteristics are then assigned to these personas. First, we segment the common interests of the real user base and assign them among the personas. We then select cultural interests from the *Eventguide*-App and assign them to the personas. Further characteristics, such as "Occupation", "Education", and "Housing" are derived from the demographics data of the user base. Other characteristics, such as "Civil status" and "Personality" are assigned evenly.

	Maria, age 26	**Gerald, age 47**	**Anton, age 32**
Occupation	Make-up artist	Middle manager	Programmer
Education	Abitur	Diploma	Master
Civil status	On-Off relationship	Married, 2 children	Single
Housing	Shared apartment	House in suburbs	Apartment in city
Interests	Green living, Asia-travels, yoga	News and politics, investments, alcohol (gin)	New technologies, mobile gaming
Culture	Theater, parties	Exhibitions, classical concerts, opera	Metal & rock concerts, museums
Apps	Instagram, Spotify, Pinterest, Amazon	Audible, News, Stocks	Netflix, Bitcoin, Games
Personality	Extroverted, empathic	Organized, factual, family man	Introverted, analytical
Motivations	Socialize on weeknights, flaunt theater knowledge	Get distracted from work on weekends, attend family events	Become part of a group, gain knowledge, date women

Table 4.3.: *Personas*

4.2.3. Usage

The desires of our personas can be derived by gaining deeper insight into their backgrounds and motivations. Similarly, the functional requirements can also be derived. For example, Maria would likely want to create her own theater events, so that she can flaunt her knowledge. Or Gerald could desire both adventurous hiking tours and family events. Finally, Anton would want to overcome his social inhibitions in a gentle way. In conclusion, requirements should provide value for at least of our personas.

4.3. Existing Approaches

This section presents existing approaches to both social matching and event recommendation, as this thesis combines these aspects. We first create a list of social matching services, such as *Tinder* or *Bumble*, by analyzing which are most popular by usage. This list is complemented by a second list of event recommenders. Both lists will include the same website or app if it matches people to people as well as to recommended events.

All services on both lists are evaluated for which features they support (reciprocal matching, for example, or the extent to which they integrate events). This evaluation shows that features can be categorized in three dimensions: event recommenders, event-based social networks, and people recommenders.

Event recommenders focus on recommending commercial events, for example plays and concerts. To improve the quality of their recommendations, most of these apps integrate external data sources. One could, for example, integrate *Spotify*, so that bands that users have listened to are more likely to be recommended when they have a performance coming up. Tickets for these concerts can then be bought directly from the event recommender app or website. Examples for such services include *Eventguide* [25], *Eventbrite* [24], and *Bandsintown* [11].

Event-based social networks focus on non-commercial events that are created by users. Examples for such events include cooking in a group or hiking together. Unlike with event recommenders, these social networks include a people dimension: A user might be urged to join a cookout with the information that eight other people with publicly accessible profiles will also participate. If a user is interested in this event, then he can sign up for the event without anyone's permission. This means that the recommendation goes in one direction only, i.e. it is not reciprocal. Recommending events is, however, only one aspect of these services; the other is social networking, such as forums, messaging, and photo sharing. These features help build communities of like-minded people. Examples include *Meetup* [40] and *Münchner Singles* [43].

People recommenders recommend people to people, not only for dating but also for employment, mentorship, and in other domains. In all cases both parties must approve of each other; making people recommenders reciprocal. Reciprocal recommenders face unique challenges, as explained in Chapter 2. Further challenges may arise when recommendations are group-to-group. All people recommenders support at least 1-to-1 recommendations. Examples include *Tinder* [62], *Bumble* [15], and *Lovoo* [35].

This thesis optimizes an event recommender by extending it with social features. We improve the recommendation algorithm to factor in the people that want to visit the events. To achieve this, we include facets of all three categories mentioned above. We needed to consider the context of event recommendation, the social aspects of event-based social networks, and the challenges of reciprocal recommenders.

4.4. User Stories

In this section, we define functional requirements for our implementation. We first describe our approach, brainstorm desired features, and define the scope of this thesis. We then create a list of desired features in the form of user stories.

4.4.1. Approach

Desired features were first brainstormed [47] in the previously mentioned kick-of meeting. Participants have received relevant background knowledge, namely what users are interested in and which other approaches exist. Following these two analyses, each person was to ideate features of an app that fulfills the desires of our personas. These features were first brainstormed individually and then recorded with markers on sticky notes and pinned to a whiteboard. Each feature was then discussed in the group, such as if it could be grouped with other features and if it was part of a minimum viable system or rather out of the scope of the project.

This scope of the project was further defined in the form of a context diagram [56]. This diagram shows which features our implementation is responsible for by defining the flow of data between our recommender system and its adjacent systems. This definition of the system boundary also defines the scope.

After the group meeting, the next step was to create user stories. We first defined the roles that people can take when interacting with the system by including the background of the list of desired features and the context diagram. We then created the user stories and prioritized them, so that we know which ones are required for a minimum viable system.

4.4.2. Desired Features

Figure 4.1 presents the desired features; these were grouped in six categories:

First, an **opt-in** feature to our social matching system should be included, as we extend a running system whose existing functionality must not be altered.

Second, **onboarding** will be the first barrier to enter our social matching features. A good onboarding experience requires that a new user will enter his private data willingly, so that he can receive our recommendations.

Third, **privacy** features are valued by many users, for example our personas Gerald and Anton, who both want to disclose their personal data only when necessary. Additionally, Maria would like to be able to block people she feels uncomfortable with from contacting her. Users should feel safe when using the app; one could therefore implement a feature where users can rate one another after they have met in person.

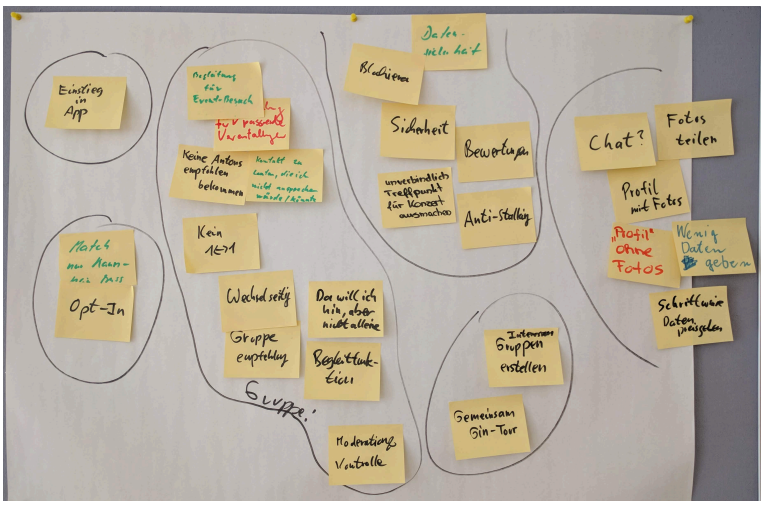


Figure 4.1.: *Brainstorm of desired features (in German)*

Fourth, **user-created events** are of interest for Maria, who would like to create her own event to flaunt her theater knowledge, or for Gerald, who would like to join a gin-tasting tour. An aspect of user-created events is that they are usually smaller and more social compared to commercial events.

Fifth, **social networking** features, such as messaging, forums, and photo sharing, improve the connections that users form within the app. However, these features also impact other desired features, such as data privacy.

Sixth, **people recommendation** features form the core of our implementation. Users may have a wide range of preferences as to which people are the right people to be recommended. Some might prefer 1-to-1 companionship recommendations, while others prefer larger groups. Maria might only want to meet people who are similar to her, while Anton wants to meet people he would not get to know otherwise. While people recommenders should be reciprocal, for larger groups this becomes impractical, as each group member would have to confirm all new members. For this purpose, we introduce a group leader role, who moderates the group and decides who can join. Furthermore, we only focus on smaller groups of up to five people in this thesis.

4.4.3. Scope

During the discussion on the desired features, a context diagram was also created to define the scope of the social matching system, as seen in Figure 4.2. The context diagram describes the system's boundaries by describing the flow of data between it and the adjacent systems. Examples for adjacent systems include *Facebook*, which

provides predefined nicknames and profile pictures to enhance a user's profile; or *München Ticket* and *in münchen*, which provide data about upcoming events. Most importantly, the user himself is an "adjacent system" who inputs his data to create a profile to receive both event and people recommendations.

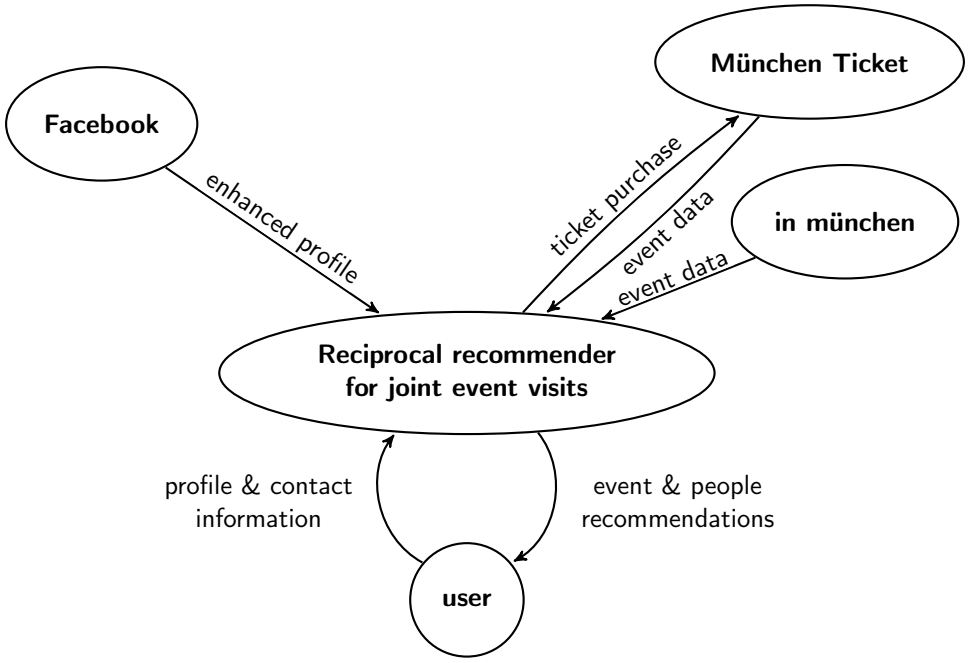


Figure 4.2.: *Context diagram*

4.4.4. Results

The group meeting resulted in a list of desired features grouped in six categorizes, and in a context diagram that defines the scope. Both results are taken into account to derive user stories. The first step was to define the following four user roles:

- **Event-seekers** are all people who use the *Eventguide*-App.
- **Users** are the subset of event-seekers who join the social matching system.
- **Group leader** of a group is the user who created it
- **Group members** of are group are all users who joined it.

It is important to note that a user can lead and join multiple groups; a group, however, can only have one leader.

We then define the following 14 user stories:

US1 As an event-seeker, I want to join the social matching system, so that I can visit events with other people, knowing that for these features my personal data will be shared.

US2 As a user, I want to share a minimum amount of personal data with the social matching system, so that my personal data is protected.

US3 As a user, I want to know what my personal data will be used for, so that I feel safe in using the app.

US4 As a user, I want to create a profile, so that I can take part in the social matching system.

US5 As a group leader, I want to create a group for an event, so that other users can join me in visiting an event.

US6 As a user, I want to see which events have group that are open for new members, so that I know which existing groups I can join.

US7 As a user, I want to see the members of an existing group, so that I know who I would be visiting an event with, if I were to join that group.

US8 As a user, I want to apply for membership in a group, so that I can join it.

US9 As a group leader, I want to accept or deny applications to my group, so that I can moderate the members of my group.

US10 As a user, I want to be notified of whether my application to a group was successful, so that I am informed about the status of my group memberships.

US11 As a group leader, I want to filter possible group members according to gender and age, so that unwanted applications are avoided.

US12 As a group member, I want to contact other members of my group, so that I know where and when to meet.

US13 As a user, I want to block other users, so that I can avoid unwanted contacts.

US14 As a group member, I want to see statistics and user ratings of other members of my group, so that I can feel safer when meeting them.

Of these 14 user stories, the stories 4–9 define the core functionalities of a minimum viable system, allowing users to join groups for events. All other user stories are desired to bring the system to production, but are not necessarily needed for a prototype or for an evaluation of the prototype. Therefore, this thesis puts a focus on user stories 4–9, i.e. the minimum viable system. However, while we put a focus on the minimum viable system, we do implement some of the other user stories (e.g. 1, 2, and 3).

4.5. Wireframes

This section presents wireframes of the user interface to be implemented in Chapter 5. According to Brown, "[wireframes offer] a simplified view of what content will appear on each screen of the final product, usually devoid of color, typographical styles, and images" [14, p. 166]. They are low-level templates for how a screen in the app will look like, how information is placed and prioritized on a screen, and how users can navigate between the screens [27, pp. 128–131]. In this section, we first create such wireframes and combine them in an interactive prototype of the system. We then explain the user interface by explaining each feature and how a user can navigate from screen to screen.

4.5.1. Interactive Prototype

If we want to create wireframes, we need to derive their content from user stories, because user stories describe the functionality and because we want to design a system that is easy to use. User stories 1–14 suggest limitations and features for the creation of wireframes. When user story 1 demands that our implementation of a social matching system must be optional to use, then this limitation forces us to not alter the existing functionality of the *Eventguide*-App. Other examples include user stories 4 and 5 which deal with the creation of profiles and groups, suggesting that these features must be represented by according screens.

Wireframes can be connected to each other to create an interactive prototype by linking specific areas from one wireframe to another. For example, if screen A had three buttons 1, 2, and 3; then clicking the area occupied by buttons 1 and 3 could show screen B, while button 2 would presents screen C. These interactive prototypes are used to explore the flow between screens.

We used *Balsamiq Wireframes* [10] to design 36 wireframes and linked them to create an interactive prototype. We adhered to iOS design guidelines, as we were to extend the preexisting iOS-Version of the *Eventguide*. This lead to some limitations, we could for example not alter the *iOS Tab Bar* [9] with five elements, or the existing layout of the start page, recommendations, and event details screens. When the interactive prototype was finished, we exported it as a *pdf*-file with 36 pages. A tester can open this file on a mobile device in full-screen mode, mimicking the intuitive behaviour of a native app.

The interactive prototype was created in an iterative process, by gathering feedback from two testers (both developers of event recommenders who were in the kick-off meeting). Stakeholders had different notions on how the desired features, brainstormed on sticky notes (see Figure 4.1), would be implemented and the interactive prototype was very valuable in gaining a consensus on how social matching will work. By overcoming these misunderstandings, we also sharpened our focus on the implementation.

4.5.2. User Interface

Figure 4.3 details six of the 36 wireframes from the interactive prototype. This collection of six screens covers functionality from user stories 2–9 and 12. We therefore cover the functionality of the minimum viable system as it consists of user stories 4–9, representing the core functionality of the system. In the following, we present this core functionality and flow from screen-to-screen further:

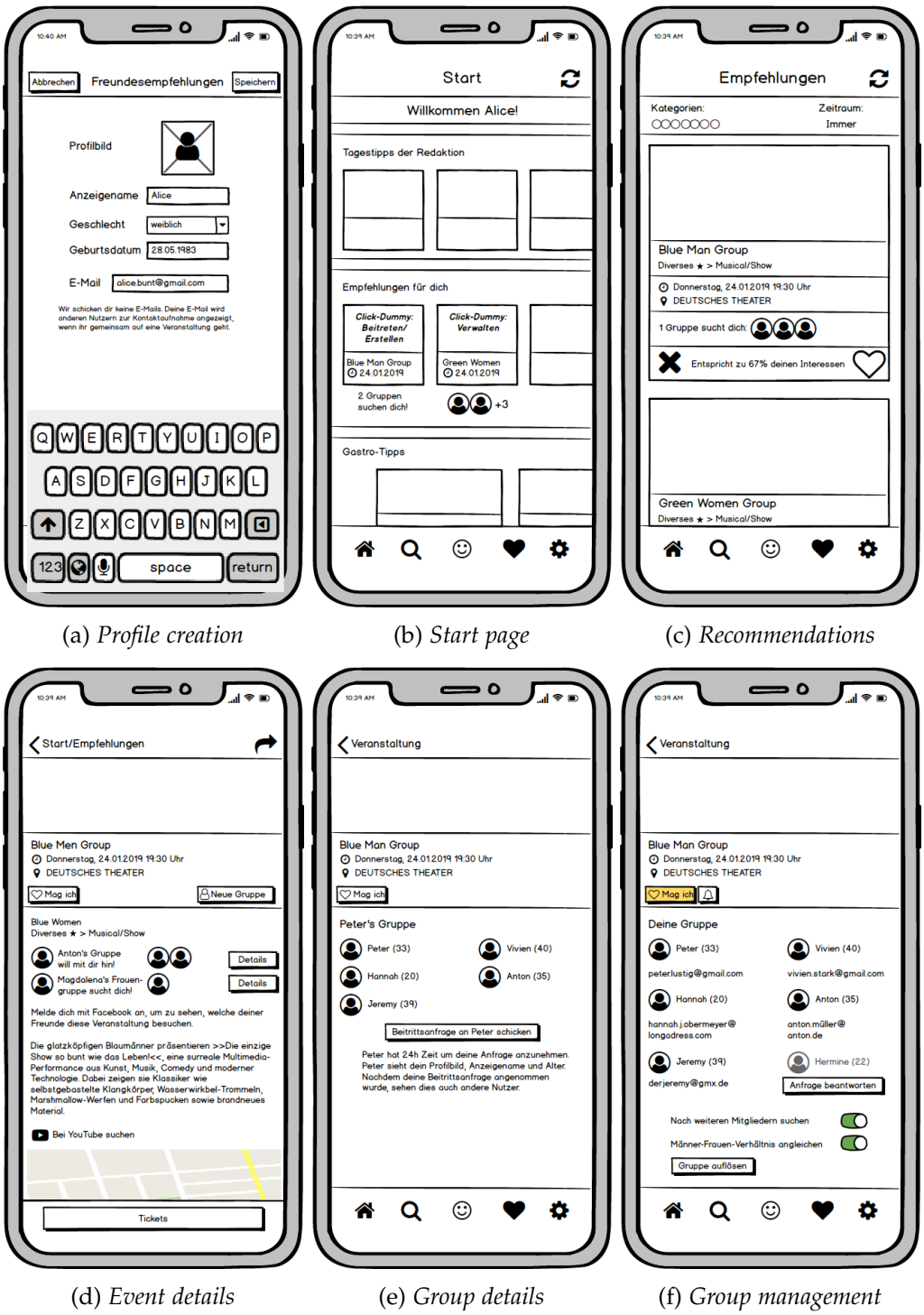
Profile creation (Figure 4.3a): A user must first create a profile for the social matching system. The user needs to share only the data necessary for social matching, including a nickname, gender, birthdate, and a profile picture. At the same time, he receives an explanation on how this data will be used, such as to whom it will be shown, therefore, the profile creation screen covers user stories 2, 3, and 4.

Start page (Figure 4.3b) and **Recommendations** (Figure 4.3c): After a user has created a profile, he will see either the start page screen or the recommendations screen. Both present upcoming events with an appropriate image, and state the time and place. These screens are preexisting from the current app and we must keep the look and feel of the user interfaces. We extend these screens by adding information about groups below each event, if for example an event had only one group, then we present the profile pictures of those group members. If there are multiple groups, this would be shown in a corresponding text label, so a user can see which events have which groups. Therefore, the start page and recommendation screens both cover user stories 6 and 7.

Event details (Figure 4.3d): If a user is interested in an event, he can tap on it to open this event details screen which presents a description of the event, shows its location on a map, and links to both *YouTube* and *Facebook*. This screen is also preexisting from the current app and we extend it by adding a button which allows a user to create a group for the event. We extend it further by showing which groups already exist, presenting the members of these groups. This covers user stories 5, 6, and 7.

Group details (Figure 4.3e): A user can tap on an individual group in the event details screen to arrive at this group details interface. He can see more information about the group members, such as their name and age. He can also apply to the group and is presented with a text label on what this means and how his personal data will be shared. This covers user stories 3, 7, and 8.

Group management (Figure 4.3f): A group leader can manage his own group when he opens the group details screen of his own event. He can then see the contact details of each member, currently in the form of an email address. One can imagine more convenient ways of contact, such as a built-in group chat, but such research was not in the scope of this thesis. Group leaders can also accept or deny applications to the group, as well as manage to which extent the recommendation algorithm should promote the group to further users. This covers user stories 9, 11, and 12.



(a) *Profile creation* (b) *Start page* (c) *Recommendations*

(d) *Event details* (e) *Group details* (f) *Group management*

Figure 4.3.: *Wireframes extracted from the interactive prototype (in German)*

5. Implementation

In this chapter, we extend the pre-existing event recommender to create a reciprocal recommender for joint event visits. For this purpose, we implement the features described by the user stories and envisioned by the wireframes.

Architecture

First, we explain the architecture of the pre-existing system:

Herzog implemented a client-server architecture where the clients are instances of mobile applications [29]. These mobile device clients can provide contextual data, such as location data, which can be input into the recommender's algorithm.

The server holds the database of events and provides interfaces for the clients to connect to. When requested by a client, the server computes a list of recommended events, using a hybrid algorithm (as explained in Chapters 2 and 3).

This architecture is presented in Figure 5.1:

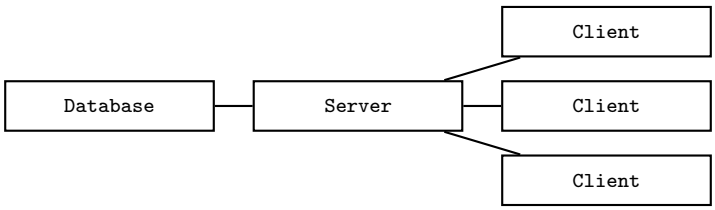


Figure 5.1.: *Client-server architecture of pre-existing app, adapted from [29]*

Tools and Technologies

Second, we present the tools and technologies used in this thesis:

The pre-existing iOS app was extended in Swift 4 using XCode 10.1 on macOS 10.14.2, because the pre-existing app was already implemented in Swift. We used the CocoaPods dependency manager [20] and integrated the *TOCropViewController* [46] framework; we also integrated the *Eureka* framework [65]. These integrations will be explained in Section 5.1.

The pre-existing server was extended in Java 8 using Eclipse 4.11. We use Hibernate [53] to map object-oriented Java classes to a relational database (MariaDB [38]). We also use Apache Mahout [6] for collaborative filtering tasks. Both of these frameworks were already integrated into the pre-existing system by Herzog [29].

Finally, the source codes for both app and server are stored in GIT repositories. We used SourceTree 3.1 [60] to interface with these.

Chapter Outline

In Section 5.1, we first describe the app and present new features from the user's perspective. We also go into detail about how these features were implemented and explain the layout and design considerations that arose.

Section 5.2 then presents how the server was enhanced to support the expanded app. Both user data and group data are stored in the database, and new API endpoints are available for the app to connects to.

In Section 5.3, we extend the pre-existing event recommender algorithm by integrating a social matching algorithm. The enhanced recommender should emphasize events that a user could jointly visit in a group with others.

Finally, in Section 5.4, we summarize the contents of this chapter.

5.1. App

This section describes how we enhanced the pre-existing app with social matching features. These features are described by the user stories (see Subsection 4.4.4) and the wireframes (see Figure 4.3). Figure 5.2 presents how we can navigate these features, i.e. between the components and screens used for social matching. It can be seen that features are split into two paths: A user must first create a profile and can then join groups for events. Both paths, profile creation features and group features, are explained in the following: from the user's perspective, layout and design considerations, and further implementation details, such as which frameworks were integrated.

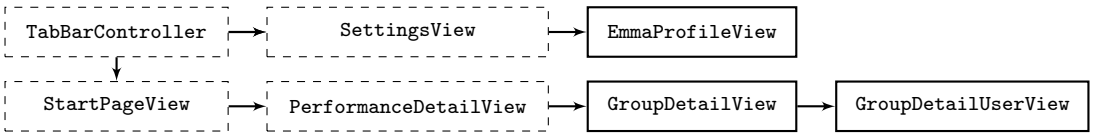


Figure 5.2.: *Navigation through components relevant to social matching; those with dashed borders are pre-existing, those with thick borders are new*

5.1.1. Profile Creation

If a user, for example our persona Maria, wants to join the social matching system, then she must first create a profile via the settings tab of the app. She then sees the locked profile form shown in Figure 5.3a. Maria can unlock this form to enter her personal information, including her profile picture, name, gender, birthdate, and email address. These five attributes form our user model: the profile picture, name, and birthdate are used to present a more meaningful profile to other users; the gender helps optimize the reciprocal algorithm detailed in Section 5.3; and the email address allows group leaders to contact the members of their group.

Editing

Maria can unlock the form and enter her data by tapping the "edit"-button, as seen in Figure 5.3b. She can then select a profile picture from the smartphone's photo library or take a picture with the camera, permitting the app the respective access if necessary. Maria can then zoom, rotate, and crop the selected image.

The default birthdate of the date picker interface [8] is set to January 1, 1980. We evaluated the age of current users in Section 4.2 and a default age of 39 years matches well with the user demographics. We therefore reduce the required scrolling distance and Maria does not have to scroll far to arrive at her year of birth.

The app also checks whether the entered information is valid. This means that a gender must be set, a user must be 18 years old, and a valid email address format must be used. If the data is invalid, then a suitable error message informs the user.

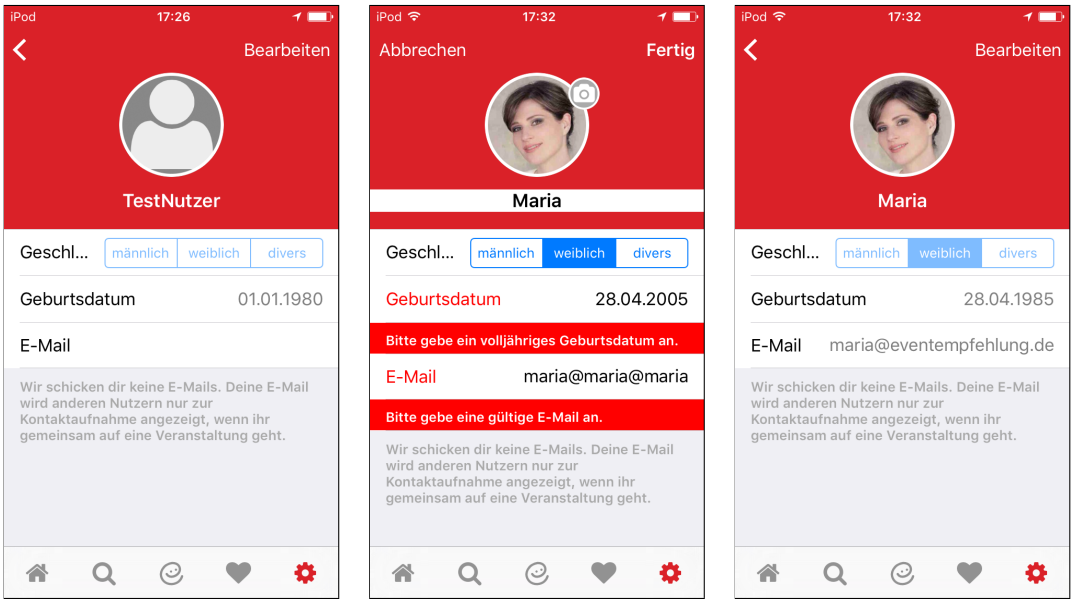
A text label at the bottom of the screen also informs the user about how her personal data, such as her email address, will be used.

When Maria taps the "save"-button and the entered data is valid, then her profile is created, as seen in Figure 5.3c.

Implementation

The *Eureka* framework [65] allows developers to rapidly implement form interfaces for user input. *Eureka* is widely used: It has more than 9.000 stars on Github [66] and more than 15.000 apps in the Apple App Store use it [65]. *Eureka* is flexible, well-documented, and supports features like validations; so we selected it as the underlying framework for the profile creation interface. Listing 5.1 provides a code example, where we define a form with four rows, including an email address row that validates whether input data conforms to email address format guidelines.

We also integrate the *TOCropViewController* framework [46] into our app, allowing us to zoom, rotate, and crop profile pictures.



(a) *Locked profile* (b) *Invalid editing* (c) *Saved profile*

Figure 5.3.: *Profile creation with validity checking (in German)*

```
1  func defineForm() { // This is an Eureka form
2      form +++ getSection()
3      <<< getGenderRow()
4      <<< getBirthdateRow()
5      <<< getEmailRow()
6      <<< getInfoRow()
7  }
8
9  func getEmailRow() -> EmailRow {
10     return EmailRow() {
11         ...
12         $0.add(rule: RuleEmail())
13         $0.validationOptions = .validatesOnDemand
14     }
15     ...
16     .onRowValidationChanged { cell, row in
17         self.getValidationErrorRow(row: row, tag: tag, text: "Bitte gebe
       ↪ eine gültige E-Mail an.")
18     }
19 }
```

Listing 5.1: *Eureka form that validates email addresses*

5.1.2. Group Features

After Maria has created her profile, the app's start page provides recommendations for events. Figure 5.4a presents such a start page where the event "*Jazz im Hofspielhaus*" has two groups and the event "*Malis Schurz*" has three groups.

When Maria taps on the first event, she sees the event details shown in Figure 5.4b. This event details screen includes a description of the event and a call-to-action button urging Maria to buy tickets. The screen also shows that two groups are looking for more members: one group created by "*Anna*", and another lead by "*Peter*". Maria can join one of these groups by tapping the group to open the group details screen, seen in Figure 5.4c. This details screen displays the group's members and includes a button for applying to the group. The group leader can then accept Maria's application.

Maria could alternatively start her own group by tapping the button on the upper right of the event details screen. Her group would then be presented to other users and she would have to accept or deny their applications.

Start Page

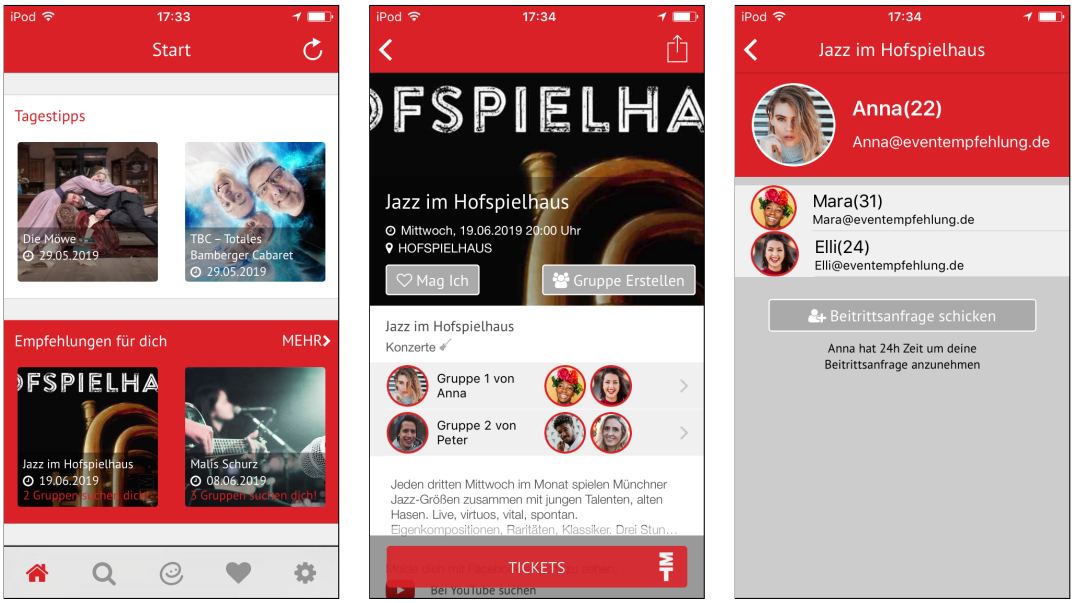
We designed the start page by adapting the vision of its wireframe (Figure 4.3b): If an event had exactly one group, then the profile pictures of that group's members would be shown; and if it had multiple groups, then a text label would be displayed. We implemented this version of the start page and ascertained that it was inadequate, as the start page became overloaded by images. In other cases, events with no groups produced gaps in the start page interface which negatively impacted the look and feel of it. We thus decided to only display text labels, such as "*2 groups looking for you!*".

Another entry point for events is the recommendations screen, as seen in the wireframe of Figure 4.3c. An extension of this screen with group features was not in the scope of this thesis, as the start page already served as our entry point.

Event Details

We implemented the event details page according to its wireframe (Figure 4.3d). A button on the upper right creates new groups; this addition to the pre-existing interface was easily accommodated. We split the event details between the genre text label and the event description to insert event groups. These groups present the name and profile picture of the leader; and up to two profile pictures of other members. A "+X" label will also be displayed if there are more members in a group than can be fit into the display row, where X is the number of members that could not be shown.

As required, we did not alter the pre-existing functionality of the event details interface, except for users having to scroll down to see elements that were shifted down.



(a) *Start page* (b) *Event details* (c) *Group details*

Figure 5.4.: *Extending the app with group features (in German)*

Group Details

We significantly altered the layout of the group details screen compared to its wireframe (Figure 4.3e): the upper one-thirds of the screen was to display the group's event, and the bottom two-thirds was to display the group's members in an arrangement of two columns. These plans were not executed: first, because a layout with two columns did not fit; second, because it was too complicated to integrate new interface elements into the pre-existing interface. Instead, we segued into a new screen which arranged the group members' details in a single column.

This new screen shows the group leader at the top and the members below. A button at the bottom can handle various functions: if a user is not a member of the group, then he can apply to it; if he is not yet accepted, then he can retract the application; and if he is a member, then he can leave the group. For each function, a label below the button presents additional information. An application to the group is automatically rejected after 24 hours, for example.

This group details interface is also used for group management if the leader of the group accesses it. Applicants to the group are then shown as greyed-out members in the list of members. A group leader can then tap the applicant to open a dialog box, allowing him to accept or deny the application.

Implementation

We integrated these screens – start page, event details, and group details – into the pre-existing app while keeping the look and feel of the design. We, for example, matched the the pre-existing colour scheme by focusing on a specific shade of red (`0xDA2128`), and implemented this shade on profile picture borders and on the group details page. We reused existing elements, for example in the design of the "*create groups*"-button on the event details page. We also focused on correct fonts, matching font-sizes, and the use of icons from the integrated *Fontawesome* icon set. However we were limited in our choice of icons because this icon set was outdated.

The app extends `UITableView` in the class `SelfSizedTableView` to allow self sizing table views, as defined in Listing 5.2. `SelfSizedTableView` can dynamically change its height to display groups on the event details screen, or present various amounts of members on the group details page. The interface elements below the self sizing table view can then dynamically move in their vertical position, so that the margin between them and the table view stays the same.

```
1  class SelfSizedTableView: UITableView {
2
3    override func reloadData() {
4      super.reloadData()
5      self.invalidateIntrinsicContentSize()
6      self.layoutIfNeeded()
7    }
8
9    override var intrinsicContentSize: CGSize {
10     return CGSize(width: contentSize.width, height: contentSize.height)
11   }
12 }
```

Listing 5.2: *Self sizing table view, adapted from [12]*

The pre-existing app integrates the *SwiftHTTP* [19] framework to communicate with the server. This framework is outdated and with the release of Swift 4 in 2018, the parsing of JSONs became much easier in the app. We decided to decouple the app from *SwiftHTTP* by implementing our server communication natively.

In conclusion to the implementation of the app, we implemented the minimum viable system defined by user stories 4–9 in Subsection 4.4.4. Further user stories were not in the scope of this thesis, such as notifications about the status of an application, filters for group members, and user ratings.

A user can now interact with the app, but for the app to function it must be connected to a server. In the following, we link the new functions of the app to the server.

5.2. Server

The pre-existing server provides the functions of an event recommender system and is implemented in Java. It was first implemented by Herzog [30] and then developed further by our industry partner. We enhance this event recommender with social matching features: The server must first enable users to create a profile for social matching; it must then enable users to create groups for event performances; and also allow users to join existing groups. The event recommender algorithm must also be adapted to take the groups into account, this will be detailed in Section 5.3.

The server consists of both the database and the RESTful web service [29]. The database stores the new data, for example a user's profile, or who is in which group for which event. The web service provides an API for the app to connect to, so that the app's new features function correctly.

We first go into detail about the database: first, we explain how its schema of tables was extended; second, we describe how the database is accessed by the web service.

Afterwards, we describe how the web service provides profile creation features and group features to the app. We specify the relevant endpoints of the RESTful API and present further implementation details.

5.2.1. Database

The pre-existing server uses a *MySQL* relational database management system (RDBMS). Herzog selected a relational database in his initial implementation because the event recommender stores mainly structured data, such as the artist, genre, or location of an event [29]. RDBMS' also have decades of production experience and developers are much more likely to be experienced with it, compared to non-relational (NoSQL) database management systems [44].

We store enhanced profile data and group data in the pre-existing RDBMS, as the data to be stored is well-structured, such as the name, gender, and birthdate of a user.

Database Schema

Figure 5.5 presents the database schema of tables relevant to social matching.

The pre-existing `USER` table is first linked to a new `USER_DETAILS` table which stores a user's personal information. We defined the `USER_DETAILS` table in collaboration with our industry partner: We included the user's name, gender, birthdate, email address, and profile picture; and limited the size of profile pictures to 1MB by selecting *mediumblob* as the data type. We decided to decouple the pre-existing user data from the personal information, because the social matching system is an optional component and not all users will want to enhance their profile by entering their private data.

Afterwards, we defined new tables on our own. We first associated the existing `PERFORMANCE` table to a new `GROUP_EVENT_ATTENDEE` table, which was supposed to be named `GROUP`, but this name is on the list of reserved words and is therefore forbidden as a table name in *MySQL*.

Users with an enhanced profile could then be associated to event groups via `GROUP_MEMBERSHIP`, which holds foreign keys to both `USER` and `GROUP_EVENT_ATTENDEE`, making these relationships n-to-1. If a user u has no association to a group g, then there will be no entry in `GROUP_MEMBERSHIP` where `USERFK` is u and `GROUPFK` is g. If there is, then user u can either be the leader, a member, or an applicant to the group g; this role is stored in the `ROLE` column.

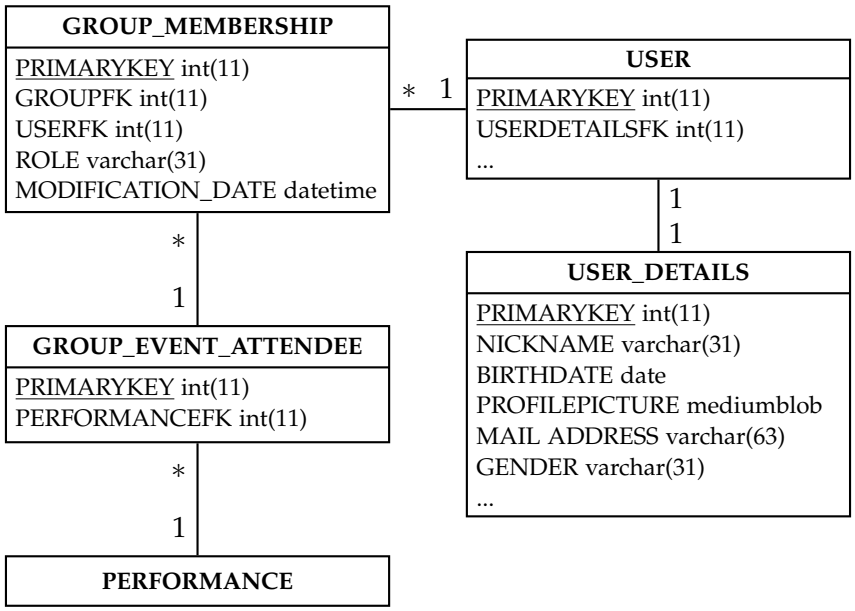


Figure 5.5.: *Database schema of tables relevant to social matching*

Database Access

The pre-existing server is implemented in Java, as explained in the introduction to this section. It uses *Hibernate* [53] to map object-oriented Java classes, also known as business objects, to the relational database. We also use *Hibernate*, since it is already integrated and since we also need to access the database from the web service. We create three new business objects (`UserDetails`, `GroupEventAttendee`, and `GroupMembership`) and map them to the respective database table.

The server also implements the *Data Access Object* (DAO) pattern which "decouple[s] the persistent storage implementation from the rest of [the system]" and "provide[s] a uniform data access API" [5]. We extend this pattern by developing a `GroupDAO` interface and a `GroupDAOImpl` class that implements this interface. Other components of the recommender system, for instance the recommender algorithm, can then load data from the database by calling methods of `GroupDAOImpl`.

The reciprocal algorithm must compute a number of values, this is further detailed in Section 5.3. One such computation is to create a list of all people a user has previously been in a group with; this list is then used to calculate the ratio of male users to non-male users. This function is provided by the `getMaleRatioOfPreviousGroupsFromUserPK (Integer userPK, ...)` method of `GroupDAOImpl` which takes a user's primary key as input and returns a ratio between 0% and 100%. The example SQL query executed for this computation is detailed in Listing 5.3 for a user with a primary key of 25222.

```
1 SELECT SUM(IF(gender='MALE', 1, 0))/count(*)
2 FROM (SELECT h.userfk, d.gender
3    FROM (SELECT g.groupfk
4       FROM GROUP_MEMBERSHIP g
5       WHERE g.USERFK=25222) f
6    INNER JOIN GROUP_MEMBERSHIP h ON f.groupfk=h.groupfk
7    INNER JOIN USER u ON u.primarykey=h.userfk
8    INNER JOIN USER_DETAILS d ON u.userdetailsfk=d.primarykey
9    WHERE h.userfk!=25222) x
```

Listing 5.3: *Example SQL query*

To summarize the database details: We extend the pre-existing database with three new tables, use the same technologies to access these three tables from the web service, and use SQL queries for more complex calculations.

5.2.2. Profile Creation

With the database connected, the server can then create user profiles by storing the user's personal data in the database.

The implementation of this feature was carried out in collaboration with our industry partner. We first defined the requirements for this feature on our own, by modeling the database shown in Subsection 5.2.1 and by describing the needed functions. The implementation itself was then carried out by our industry partner, since the relevant `UserResource` class is also tightly coupled to other projects under development.

`UserResource` encapsulates the resources so that clients can request and receive data about users. Endpoints in this class access the database via `UserDAO` and the data is then returned as a JSON, if necessary first transformed by `UserDTO`.

Table 5.1 presents the five API endpoints relevant to profile creation:

First, we create or update a profile by posting the required personal data to the `/user/update/details` endpoint. We encode the input as *multipart/form-data* [39] to upload the profile image file; this is more suitable to the alternative *application/x-www-form-urlencoded* because the image is a binary file and may be up to 1MB in size.

Second, we load a user's data by passing his ID to the `/user/findByID` endpoint, specifically the app's `identifierForVendor` property [7]. The endpoint returns a JSON containing a user's entire set of data including his enhanced profile (if available), but without his profile picture. Analogously, the same data is returned by passing the user's primary key to the `/user/findbyPK` endpoint.

Third, the profile picture is loaded separately due to its file-size. It can be loaded by passing the active user's ID to the `/user/retrieveProfilePicture` endpoint, this conforms to pre-existing code design guidelines. To load profile pictures of other users we pass a primary key to the `/user/retrieveProfilePictureByPK` endpoint.

URL	Method	Params	Function
/user/update/details	POST	userID profilePicture nickname gender birthday email	Stores enhanced profile
/user/findByID	GET	userID	Loads user data
/user/findByPK	GET	userPK	Loads user data
/user/retrieveProfilePicture	GET	userID	Loads profile picture
/user/retrieveProfilePictureByPK	GET	userPK	Loads profile picture

Table 5.1.: *Resources of `UserResource` relevant for profile creation*

5.2.3. Group Features

Table 5.2 presents the four API endpoints that manage groups. A user can create a group for a performance and thus becomes the group's leader. Other users can apply to this group; the leader can then either confirm or deny these applications.

These functions are implemented in the `GroupResource` class, where each resource is called by posting the two relevant primary keys. To create a group, the client needs to pass both the leader's primary key and the performance's primary key, for example.

URL	Method	Params	Function
/group/create	POST	userPK performancePK	Creates new group with leader
/group/apply	POST	userPK groupPK	Creates new applicant for group
/group/confirm	POST	userPK groupPK	Confirms applicant to group
/group/deny	POST	userPK groupPK	Denies applicant to group

Table 5.2.: *Resources of* `GroupResource`

In the current prototype it is not possible to delete groups via a resource, nor is it possible to cancel an application or to leave a group. These functions are necessary for the app to go to production, but are not in the scope of this thesis. If these functions are needed during the evaluation (to be detailed in Chapter 6), then we can access the database manually.

5.3. Social Matching Algorithm

Users can now create and join groups, but these additional points of data are not yet reflected in the recommender algorithm. In this section, we enhance the pre-existing event recommender algorithm so that it factors in the people dimension. This social matching extension impacts an event's recommendation value, adding some percentages to events with groups. Ultimately though, the recommender algorithm still recommends events and does not recommend people.

We first present the pre-existing event recommender algorithm, which is a context-aware hybrid algorithm. For this purpose, we describe its three stages: contextual pre-filtering, content-based prediction, and collaborative filtering. We also describe how our social matching extension can be integrated into the pre-existing implementation.

Afterwards, we present this social matching extension, also in three stages: demographics, similar users, and reciprocal matching.

5.3.1. Pre-Existing Context-Aware Hybrid Algorithm

Herzog and Wörndl applied a *content-boosted collaborative filtering* (CBCF) approach to the event recommendation domain [29, 30]. CBCF is an advanced hybrid algorithm, first presented by Melville et al. [41], and it provides better recommendations compared to both pure content-based filtering and pure collaborative filtering approaches. Melville et al. further go on to state that CBCF also performs better than naïve hybrid approaches.

Herzog and Wörndl extend their CBCF algorithm so that it is also aware of the context [30]. Context-aware recommendations can be computed in three different ways, one of which is the *contextual pre-filtering* paradigm, as explained by Adomavicius and Tuzhilin [2]. In this paradigm, the set of possible events to be recommended is first filtered according to contextual preferences, such as events which are too far away from a user. From this reduced set of events, the CBCF algorithm first predicts recommendations based on content and then by collaborative filtering.

These sequential steps of the algorithm are derived from the *pipes-and-filters* architectural pattern, detailed by Herzog in Figure 5.6 [29]. Buschmann et al. define the pipes-and-filters pattern as "a structure for systems that process data streams [where] [e]ach processing step is encapsulated in a filter component [and where] [p]ipes are used to pass data between adjacent filters" [17].

The existing context-aware CBCF algorithm delivers personalized event recommendations by filtering events in three stages: first by contextual pre-filtering, then by content-based prediction, and lastly by collaborative filtering. In the following, we explain each stage in detail.

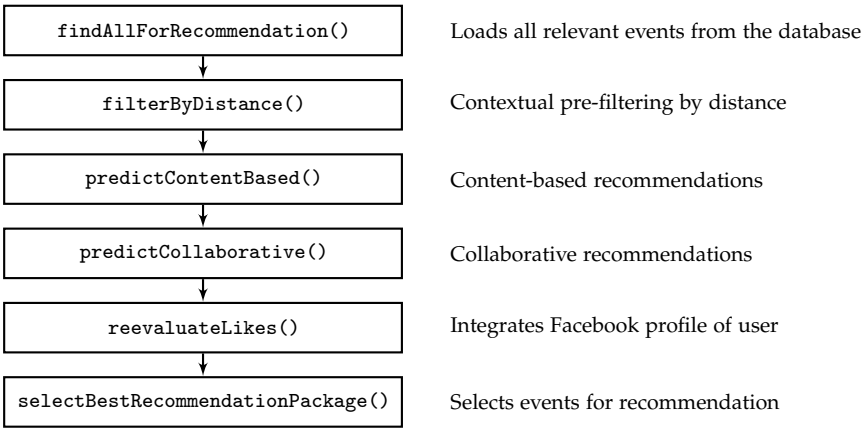


Figure 5.6.: *Pipes-and-filters architecture of the context-aware CBCF algorithm, taken from [29]*

Contextual Pre-Filtering

The first two methods presented in Figure 5.6 are `findAllForRecommendation()` and `filterByDistance()`, both of which implement contextual pre-filtering:

The first method, `findAllForRecommendation()`, filters the database of all events according to several contextual factors, including the genre of an event, both weekday and time of day of the event, and scheduling conflicts. A user can specify that he is not interested in classical music, and that he is not available on working days or in the mornings, for example. If a user has already bought a ticket for an event, then other events scheduled at that time are also to be excluded.

The second method, `filterByDistance()`, excludes events which are outside of a predefined radius from the user's location. This location can be accessed via smartphone location services or can be fixed to a specific location, the city center of Munich, for example.

Content-Based Prediction

The algorithm then predicts how similar each event is to a user's preferences using a content-based approach. One form of content-based predictions is case-based recommendation which works well for handling structured data [59]. Events are such structured data, as they have attributes like a genre, category, or venue [29].

A user can interact with the recommender by liking or disliking events, and by buying tickets. The algorithm uses these interactions to calculate a user's preferences, for example "if a user liked 90% of all recommendations for a venue, the query value q_i

for this venue is 0.9" [30].

These values q_i for each attribute are used to calculate how similar a future event e is to the user's preferences q, using Formula 5.1, as presented by Smyth [59]. Each attribute is also weighted by w, as presented in Table 5.3.

$$Similarity(q, e) = \frac{\sum_{i=1}^{n} w_i * sim_i(q_i, e_i)}{\sum_{i=1}^{n} w_i} \tag{5.1}$$

This similarity between the user's preferences q and a future event e is stored in a user-event rating matrix, but only if multiple attributes values q_i can be calulated. If a user has never rated an opera at the opera house for example, then the case-based algorithm will return the similarity value 0.5 for such an event [29]. These values stored in the user-event rating matrix are then used in the collaborative filtering stage of the CBCF algorithm.

Attribute	**Weight**
Genre	0.45
Tags	0.25
Category	0.20
Venue	0.10

Table 5.3.: *Weight of attributes for similarity assessment*

Collaborative Filtering

Collaborative filtering is applied to events whose value could not be predicted by the previous content-based prediction step and whose value is therefore set to 0.5. For this purpose, Melville et al. [41] have implemented a *user-based nearest neighbor* approach of collaborative filtering; this was then adapted by Herzog [29] using the *Apache Mahout* framework [6]. *Apache Mahout* first searches for users similar to the current user by calculating the Pearson correlation coefficient. Those users with a coefficient above 0.5 are then presumed to be in the neighbourhood of similar users; this threshold of 0.5 represents a "large effect size" [21]. The collaborative filter then improves the user-event rating matrix based on the preferences of these similar users. The existing algorithm thus rates future events with a value between 0% and 100%, using context-aware content-boosted collaborative filtering.

Integration of Social Matching Features

We integrate our new social matching features into the pre-existing algorithm by appending four new stages to the context-aware CBCF, as presented in Figure 5.7. The first stage is to load groups from the database and to attach them to their related event, because the events passed by the CBCF have no groups associated to them. After linking events to their groups, we are then able to extend the existing algorithm in three stages: demographics, similar users, and reciprocal preferences.

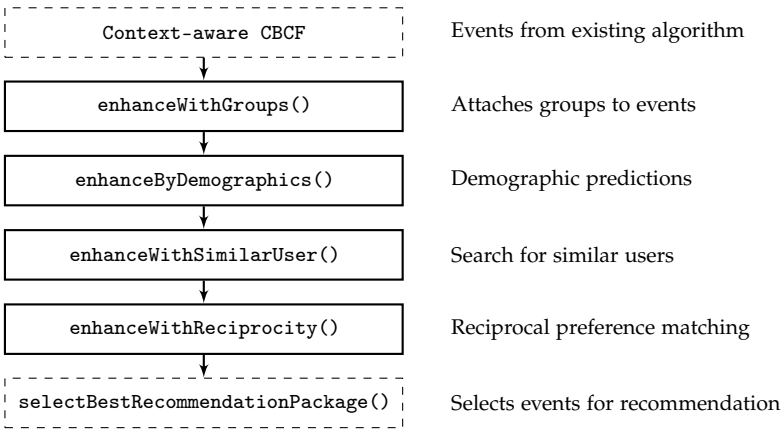


Figure 5.7.: *Extension of pre-existing algorithm with social matching features*

5.3.2. Demographics

This stage factors in demographic considerations, for example that an event with groups is more interesting than the same event without groups. We also consider that groups are more interesting to a user if the people in that group are close in age to the user.

Group Interest

We first assume that an event with groups is at least as interesting as the same event without groups. Groups are an optional feature to the event recommender and are also optional to visiting events: Users could simply ignore the groups and thus reduce the event to as if it had no groups; they would then have to make the same decision, for example whether to visit the event on their own, or which friends to invite to go to the event together with. The same reasoning applies if users find the groups not to their liking. Thus, the lower bound for an event's recommendation value with groups is equal to that event's value without groups.

We then assume that an event with groups is more interesting than the same event without groups. Users, who voluntarily joined the social matching system to be able to see other people, are indeed interested in seeing other people. If a user is interested in a group and wishes to join it, then the event this group is linked to becomes more interesting. We implement these considerations by increasing an event's recommendation value by a static amount $a = 3\%$, if that event has groups. We selected this value subjectively, it can be tweaked if evaluations show that it should be changed.

Age Range

The algorithm also considers that people are generally more interested in other people who are close in age. While our social matching extension is not specifically designed for dating, one must consider that dating is an inherent aspect of social matching.

Buunk et al. [18] researched how age preferences vary for different levels of relationship involvement, including falling in love and serious relationships. Their data suggests that "women's partner preferences follow a simple rule: they prefer men around their own age, with a range from slightly younger to slightly older". They go on to show that men's age preferences are not so straight-forward: 50-year-old men consider 30-year-old women as the mean minimum age for falling in love with. 30-year-old women, however, do not reciprocate these advances: The mean maximum age for falling in love with are 38-year-old men. While other asymmetries in the friendship between men and women exist [63], Buunk et al. show that reciprocating matches can mainly be created when people are close in age [18]. They further show that this range increases with age: Evaluating how young a man can be for a woman to fall in love with, the mean minimum age for acceptance is 19 for 20-year-old women, 25 for 30-year-old women, and 33 for 40-year-old-women.

We evaluate the current user demographics (see Section 4.2) to define acceptable age ranges, presented in Table 5.4. The algorithm then increases an event's recommendation value: For each group of an event, the percentage p of group members within the age range to the total amount of group members is calculated. The highest such p is then multiplied by $b = 5\%$ (also subjective) and added to the event's recommendation value.

Age	Range
18–24	± 4
25–34	± 5
≥ 35	± 7

Table 5.4.: *Age range preferences*

5.3.3. Similar Users

The next step is to integrate the users' preferences for events into the social matching system, for example if a user is passionate for opera or likes a specific rock star. We already know that users who visit an event together are interested in that event, but we can improve the quality of the event recommender by computing how those users' preferences in other events are to then match similar users.

How similar users are in their taste for events is already calculated by the existing collaborative filter, as described in Subsection 5.3.1. The Pearson correlation coefficient is a common similarity metric used in recommender systems; both Melville et al. [41] and Herzog [29] have implemented it. The similarity s between a user u and a user v is computed by Formula 5.2, where u and v have rated an event i with $r_{u,i}$ and $r_{v,i}$ respectively [23].

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{v,i} - \bar{r}_v)^2}} \tag{5.2}$$

Possible outputs of s range between -1 and $+1$, with the sign signifying whether the preferences of two users correlate directly or inversely. The magnitude then describes how strong this correlation is. We continue to use the already integrated *Apache Mahout* framework [6] for this similarity computation. We first calculate which other people are similar to the current user, using a threshold of $s = 0.3$ as this represents a "medium effect size" [21]. The algorithm then searches for such similar users: For each group, each similar users increases a group modifier q by a static base value of $c = 3\%$. This was value was selected in a subjective manual pre-trial where we found it to fit well. An event's recommendation value is then increased by the highest q, capped at 100%.

5.3.4. Reciprocal Matching

The final step is to consider reciprocal preferences. Our system aims to create people-to-people connections, but for two people to create a friendship they must like each other, i.e. this friendship must be reciprocal. Further explanations for why reciprocity is needed were already provided in Chapter 2. To sum it up, "[r]eciprocity is a core requirement for systems that are designed to facilitate the establishment of reciprocal connection between people" [48]. Therefore, this reciprocal stage of our algorithm tries to find out a user's preferences in regard to other users, so that it can better adjust the recommendation value of an event.

Pizzato et al. presented RECON [49], which is a reciprocal recommender for online dating, as explained in Section 3. We adapt the content-based reciprocal algorithm

used by RECON to our needs, because it is easy to use with our existing data. Other approaches are more optimized, including further work on RECON by Pizzato et al. [50, 51], which integrate negative preferences or use collaborative filtering; pure collaborative filtering approaches [33]; or a content-collaborative hybrid approach [4]. We do not adapt these approaches, because we neither have negative preferences in our data, nor do we have ratings for people. We could implement these features to optimize our reciprocal algorithm, but this was not in the scope of this thesis. Instead, we adapt the simple, content-based approach of RECON.

User Preferences

In our current implementation, a user like our persona *Maria* can interact with other users by joining into a group with them. Maria cannot message other users, nor can she rate other users, so the only way of gathering her preferences is to analyze her group history. We first create a set of all users that Maria has been in a group with and then aggregate the attributes of these users. When we model user attributes, we consider the explicit profile data that a user has input manually. On common dating websites, these attributes could include a user's gender, age, body shape, personality, marital status, and educational level [49]. Our algorithm only considers gender and age, since these are the only two attributes available to us.

Figure 5.8 presents an example distribution for Maria, who has been in groups with 18 different people. It shows that she prefers to be in groups with males and with users who are between 25 to 34 years old.

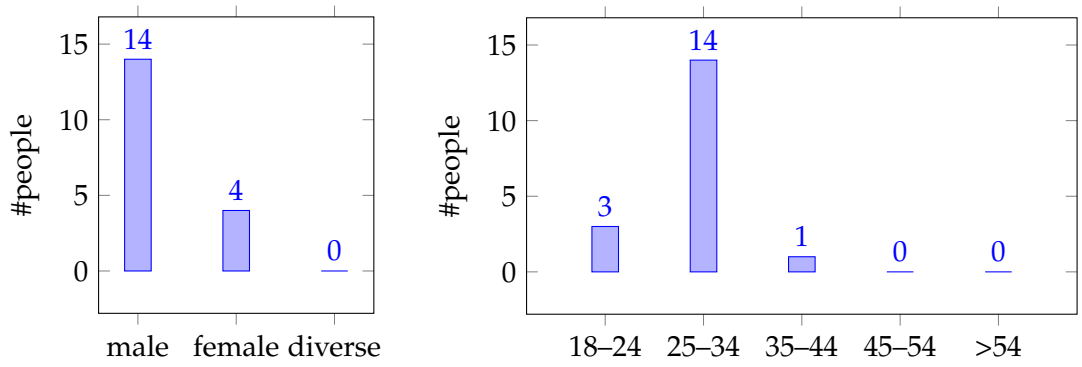


Figure 5.8.: *Example distribution of Maria's previous group members*

If Maria was 26 years old and another user *Peter* was a 37-year-old male, then their profiles U_{Maria} and U_{Peter} are presented in Table 5.5 together with their preferences

P_{Maria} and P_{Peter}. We define a user's preferences as "a group of histograms showing the number of times each value has occurred for each profile attribute" [49].

	U_{Maria}	U_{Peter}
Gender	Female	Male
Age	26	37
	P_{Maria}	P_{Peter}
$p_x, Gender$	(Male, 14)	(Male, 15)
	(Female, 4)	(Female, 26)
p_x, Age	(18–24, 3)	(25–34, 28)
	(25–34, 14)	(35–44, 11)
	(35–44, 1)	(45–54, 2)

Table 5.5.: *Example profiles and preferences of Maria and Peter*

Reciprocal Compatibility

With these preferences, we can then compute the compatibility between two people. First, we calculate the compatibility between two users using Formula 5.3 (adapted from [49]) which defines how compatible a user U_B is to U_A's preferences P_A:

$$Compatibility(P_A, U_B) = \frac{1}{2} \sum \frac{(U_B\text{'s attributes in } P_A)}{(\text{No. people in } U_A\text{'s group history})} \tag{5.3}$$

These calculations are unidirectional: Using the preferences from Table 5.5, we calculate that Maria prefers the 11 years older Peter with only 41%, while Peter prefers Maria with 66%:

$$Compatibility(P_{Maria}, Peter) = \frac{1}{2} \times \frac{14^{(Male)} + 1^{(35-44)}}{18} \approx 0.41 \tag{5.4}$$

$$Compatibility(P_{Peter}, Maria) = \frac{1}{2} \times \frac{26^{(Female)} + 28^{(25-34)}}{41} \approx 0.66 \tag{5.5}$$

We then combine these two scores to calculate a reciprocal score with the harmonic mean, as presented in Formula 5.6, adapted from [49]:

$$Reciprocal(Maria, Peter) = \frac{2}{\frac{1}{0.41} + \frac{1}{0.66}} \approx 0.51 \tag{5.6}$$

Note that Formula 5.3 assumes that a user has already been in groups, otherwise we would have to divide by zero. This cold-start problem for a new user without a set of preferences P_x is handled by setting a small compatibility of 0.001. Since we cannot take into account the new user's preferences, we instead compute which other users would prefer the new user; that is to say instead of using Formula 5.6 to calculate a reciprocal score, we simply use the unidirectional preferences.

This reciprocal score is what the reciprocal matching stage computes. For each group of an event, the reciprocal score between each group member and the user is calculated and the mean reciprocal score is defined as r. The highest such r is then multiplied by a static base value $d = 10\%$ (subjectively selected and manually tested in a pre-trial), and added to the event's recommendation value.

Limitations of this simple reciprocal matching algorithm include that a user's age is discretized into age groups, for example, ages 25 and 44 are the same distance apart as ages 34 and 35; this limitation was improved on by Xia et al. [64]. Another limitation is that the attributes of a user's preferences are weighted the same, a mitigation for this was presented by Pizzato et al. [49].

5.4. Summary

In this chapter, we adapted the requirements (both user stories and wireframes) to implement a minimum viable system for social matching. With this system finished, users can now interact with event groups: see who is joining which event, for example. We described the implementation of this system in three sections:

We first extended the app: In each subsection, we first described a new feature from the user's perspective and then explained the details of its implementation, such as layout and design considerations.

The server then followed: To store the new group data, we first showed how the database was extended. We then specified the new API endpoints of the RESTful web service, for both profile creation and group features.

Finally, we defined the reciprocal social matching algorithm: First, by explaining the existing context-aware hybrid algorithm. Second, by explaining how our social matching parts integrate into the existing pipes-and-filters architecture. Lastly, by specifying our reciprocal social matching algorithm in three stages: demographics, similar users, and reciprocal matching. These stages increase an event's recommendation value according to its attached groups.

A user is now more likely to be recommended events with groups whose members are fitting to that user. To which extent this is of value is to be evaluated in the following chapter, e.g. if users find events more interesting with groups than without.

6. Evaluation

This section evaluates the implementation described in Chapter 5:

We first explain the design of this study by formulating goals, research questions, and metrics. Then, the study objects subsection describes which events are shown in the app and how groups were composed and attached to the events. Afterwards, we extensively detail the study procedure, explaining how study participants experienced this evaluation. We then present the study results and answer the research questions; and these results are then interpreted and discussed. Finally, we explain the threats to the validity of this study.

6.1. Study Design

First of all, we detail the design of this study which uses the *Goal-Question-Metric approach* (GQM), introduced by Basili et al. [13]. We use this approach to to define research questions:

GQM is a framework for how to take meaningful measurements, e.g. for evaluations. We first define goals to understand what we want to achieve in the evaluation and what the purpose of this project was. These defined goals are then broken down into supporting research questions and we explain the rationale behind each. These questions can then be answered by a combination of metrics. Figure 6.1 presents a model of this hierarchical GQM approach:

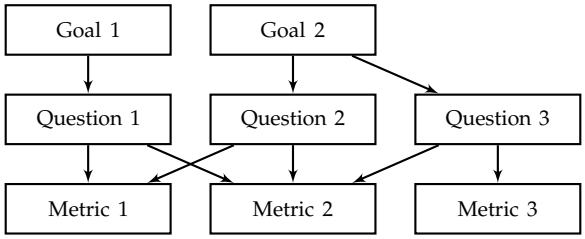


Figure 6.1.: *Example GQM hierarchy, adapted from [13]*

In the following, we describe how we apply the GQM approach:

6.1.1. Goals

Our goal was to evaluate to which extent the social matching extension implemented in Chapter 5 is of value to the pre-existing event recommender. The potential value of social matching includes various facets, that ultimately aim at the same goal: We want to increase the number of users of the event recommender.

- First, we want to make events more interesting, by allowing users to create groups for an event they want to visit with others, and by then showing these groups to other users.
- Second, we aim to improve the quality of the event recommender itself, including that its recommendations are more interesting and more diverse.
- Third, we want to deliver a social matching extension that visually fits into the pre-existing system, keeping the quality of the user interface high.

6.1.2. Questions

From these goals, we formulate the following research questions (RQ):

RQ1 **Are users more interested in events with social matching?**
This question evaluates if users find events with groups more interesting compared to events without groups. Users who are on the verge of whether to join an event might be interested to do so if they can visit the event with like-minded people. Alternatively, users could place little value in the social matching features, so that it has little to no impact on how interesting an event is. Lastly, a negative correlation could also exist: users might be less interested in an event if they see that the audience is comprised of people they dislike, feeling that they would not fit into the crowd.

RQ2 **Do recommendations improve with social matching?**
This question analyses the impact of our social matching algorithm to the event recommendation algorithm. RQ2 evaluates whether the quality of recommendations is improved when we increase an event's recommendation value by including groups. The purpose of the event recommender is to recommend events that users like, thus this question evaluates our implementation's fitness for purpose. If this was negative, then we'd have to tweak the social matching algorithm or split it from event recommendations altogether. RQ2 should correlate with RQ1, but we need to evaluate it separately: if RQ1 was true, but our implementation was bad, then RQ2 could still be false.

RQ3 **Do social matching features negatively impact the usability?**
This question evaluates whether our implementation of the social matching user interface negatively impact the usability of the app. This is important, since additional screens and views might "overload" the app. If the usability becomes worse, then a user might be less interested in social matching simply because of the user interface, thereby falsifying the results.

RQ4 **What influences do demographic factors have?**
This question evaluates the influences of demographic factors, such as age, gender, and personality. One could, for example, assume that young students new to the city are more enthusiastic to use social matching features than older people with established friend groups and family. Therefore, this question evaluates such considerations.

6.1.3. User Tests

To answer these research questions, we conduct tests with study participants. In the following, we provide an overview on the procedure of such a test:

First, we welcome the study participant and introduce him to the event recommendation domain. We then perform a setup-phase in which we create a user profile and task the tester to interact with the app. The recommender algorithm uses this setup-data to provide personalized event recommendations.

Afterwards, we use A/B-testing to compare the original event recommender with the version that extends it with social matching. We select a sequential within-subjects study design [31], meaning that each tester will experience both variants of the app. One advantage of this choice is that we need only half as many study participants, compared to the alternative between-subjects study design. The main advantage however is that individual differences between participants are mitigated, as each tester provides his own baseline A that can be compared to variant B. Disadvantages include the carryover effect, meaning that a tester might be biased by the first variant when evaluating the second. To mitigate this, we alternate the order of variants A and B for each tester.

In each scenario, we task the user to like or dislike recommended events. We also ask the user to rank these events.

After each scenario, the tester fills out a survey regarding that scenario; and when both scenarios are complete, a third survey is provided with demographic questions. Finally, we discuss the course of events with the study participant.

This study procedure is explained in detail in Section 6.3.

6.1.4. Metrics

In these user tests, we collected data so that we can measure metrics:

First, we collected survey data: We defined questions that are rated by the study participant on a 5-point Likert-scale, ranging from "disagree strongly" to "agree strongly". These questions are asked for both variants A and B, so we can compare their results. We also ask accessory questions that can help us in interpreting the results.

Second, we collected the participant's event preferences in both variants A and B to analyze the events he liked and the list of events he ranked.

In the following, we detail which metrics are used to answer the research questions:

RQ1 **Are users more interested in events with social matching?**
First, we evaluate the following survey question:
- "The events recommended to me match my interests"

We then compare variants A and B by this metric:
- The amount of likes given

RQ2 **Do recommendations improve with social matching?**
We evaluate the following survey questions:
- "The events recommended to me match my interests"
- "The events recommended to me are diverse"
- "Overall, I am satisfied with the recommender"
- "I would use this recommender again"

We also analyze the tester's preferences in the following metrics:
- The position of the first liked event
- The positions of all liked events
- The position of the most liked event
- The positions of the top 3 liked events

RQ3 **Do social matching features negatively impact the usability?**
We evaluate the following survey questions:
- "The layout and labels of the recommender interface are adequate"
- "I became familiar with the recommender system quickly"
- "I understood why events were recommended to me"

RQ4 **What influences do demographic factors have?**
We evaluate the same metrics as in RQ1 and RQ2, but instead of comparing variant A against variant B, we only use variant B and compare different demographic factors, such as age.

We use statistical testing on each metric to check if one variant has a significantly different behaviour, i.e. $p < 0.05$.

First, we could use *Student's 2-sample t-test* to compare the means, but this assumes that the 5-point Likert-scales of the survey use an interval scale, i.e. that the distance between each Likert-point is the same. Instead, we use the *Mann-Whitney U-Test* [37] which fits better for our purposes, since we assume that our Likert-scales are ordinal, i.e. the distance between each Likert-point is not guaranteed to be the same.

6.2. Study Objects

This section details the study objects that we use to answer the research questions. The app displays real events and to some of these events we attached pre-defined groups consisting of fictitious people.

6.2.1. Events

We receive events from two APIs provided by *in München* and *München Ticket* respectively. These two data sources were already integrated in the pre-existing app and together they cover a large majority of events in the Munich area: the database consists of 118,817 event performances. These events have characteristics, the relevant ones for this study are detailed in Table 6.3:

Characteristic	**Example**
Name	*Rammstein*
Image	*<Image>*
Genre	*Concert (Rock/Pop)*
Description	*Rammstein in Munich! [...]*
Venue	*Olympiastadion Munich*
Date	*June 8, 2019, 7:30pm*

Table 6.3.: *Characteristics of an event*

We use the two APIs because they are already integrated into the app. Both APIs may provide the same event in which case the server attempts to match the duplicates to provide only one event. However, while some events match, both APIs also offer a different selection of events: *München Ticket* provides events that they also sell tickets for, large concerts in big stadiums, for example. Meanwhile, *in München* provides a more comprehensive catalog of events, including smaller events such as open mic

nights in a bar. In summary, the app provides a comprehensive overview on events in the Munich area. The consequences of this large amount of performances are various: A user might discover new events more easily because of the large variety available. On the other hand, a user might be less interested in the recommender when it rarely presents events from artists known to the user, because the majority of performances are from smaller, unknown artists.

6.2.2. Groups

We need to present study participants with other users who would like to go to an event with him. It was unfeasible to match study participants to one another in the study, as we would have to perform the study with all participants at the same time. Instead, we presented study participants with the same set of fictitious users.

To create these users, we first considered which characteristics are relevant: name, age, gender, and profile picture. This is the same set of personal data that real users would have to share to create their profile, minus the email address which is irrelevant for our purposes. We also considered that a balanced gender-ratio across all age ranges is needed, to reflect the general population and potential user base. We looked back at which baby names were popular to select the names and we used attractive stock images as profile pictures. We also focused on diversity by including people of color and immigrant backgrounds. In conclusion, we created 12 users, presented in Table 6.4:

Male	**Female**
M19: Henry	F22: Anna
M25: Elias	F24: Elli
M27: Kiano	F31: Mara
M33: Peter	F38: Nicole
M43: Thomas	F40: Ulrike
M55: Johnny	F41: Anja

Table 6.4.: *Fictitious users*

These 12 users are then assigned to ten groups which are attached to five events (a user can be in multiple groups), as presented in Table 6.5. We created groups of people who are close in age and focused on different gender-ratios within the groups, including male-only and female-only groups. We attached various amounts of groups to the events and also varied in the age ranges for each event. In summary, we created groups in such a way that study participants should be interested in a few.

Event 1		Event 2			Event 3		Event 4	Event 5	
G1	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**	**G8**	**G9**	**G10**
F24	M43	M27	F38	M55	F22	M33	F41	M55	F22
M19	F40	F31	M43		F24	M27	M43	M43	M19
F22	F41				F31	F38			
M25									

Table 6.5.: *Events with groups of users*

6.3. Study Procedure

In this section, we use the study objects to conduct evaluations with study participants. We detail how we perform the evaluation and how we gather data to compute the metrics. These metrics then answer our research questions.

Each iteration of the evaluation consists of four steps: first, preliminaries; then either variant A or B; then the other variant; and finally, a conclusion.

6.3.1. Preliminaries

We first limit the app to display only concerts: this increases the performance of calculating recommendations. We reason that most people have a taste in music and can imagine themselves visiting concerts with other people. Other events, such as plays, museum tours, or sports events, are therefore not relevant to this study. One could, for example, argue that the experience of hiking together is more social than attending a concert, but this was not in the scope of this evaluation. We assume that our focus on concerts creates only a negligible difference in the answer of our research questions.

We then tweak the event recommendation algorithms. Previously, a diversity filter would present the user with events from different time-spans, i.e. in the next few days, next week, and next month. This filter was previously displayed in Figure 5.6 (the `selectBestRecommendationPackage()` stage), and we reduced it to simply rank events by their recommendation value. This allows us to better compare the study results, as otherwise concerts with a high recommendation value might not be displayed, because they're performed at the wrong time.

Recruitment

Finally, we recruit study participants from a target group that is diverse in age, gender, and personality, as described in Sections 4.2 and 6.2. Potential participants are rejected

from the study if they have not attended a concert in the last 2 years; they are also rejected if they have no experience with smartphones and apps. All participants are evaluated independent from one another and are also separated in location.

Introduction

We welcome participants to the study and explain what event recommenders do. We then explain that the app can recommend real upcoming concerts in the Munich area. We inform participants that we will conduct two scenarios while evaluating a few things that will not be explained further at this time, otherwise we might bias the participant. However, after the study is over, we can unveil and discuss the course of events. We also explain that a setup phase is performed before the two scenarios and that three surveys are to be filled out: one after each scenario, and one at the end. We do not explain the design of this study (goals, research questions, and metrics).

Setup

First, we create a user profile, by entering the study participant's gender and birth-year in the app. We use a default name, profile picture and email address since these are not relevant for the social matching algorithm.

Afterwards, the participant needs to interact with the system for a while, so that we can analyze their taste in concerts afterwards. We therefore ask the tester to like or dislike concerts from the *recommendations* screen of the app. This screen displays a batch of 10 concerts at a time, and uses an intuitive layout where a participant can swipe left or right to confirm whether or not he's interested. We explain that this decision may only be influenced by the following event characteristics: name, genre, location, image, and description; therefore all other characteristics are not relevant, including the time of the concert performance. We task the user to like or dislikes at least 3 sets of 10 concerts, until the highest recommendation value from the following set of 10 concerts is at least 65%. While these parameters are arbitrary, they were also validated in a manual pre-study.

After this setup, participants have created a profile and have interacted with the system. We can then begin the evaluation with either variant A or B, alternating this choice with each participant.

6.3.2. Variant A: Original

We guide the participant to the *start page* screen (see Figure 5.4a) and explain that we will be focusing on the concerts in the red "recommendations" area. The event recommender algorithm will display a batch of 10 concerts by processing the input data

from the setup phase. We task the participant to tap on each concert, so that he must enter each concert's *event details* screen (see Figure 5.4b). Using the same characteristics as in the setup, the participant must then vocally state whether he is interested in the concert or not. At the end, he must rank the top 3 of these 10 concerts. If a participant does not like at least three concerts, then he must still rank three concerts.

Afterwards, the participant is presented with a survey consisting of 9 questions that "assess the perceived qualities of recommenders such as their usability, usefulness, interface and interaction qualities, [and] users' satisfaction of the systems [...]" [57], using the *ResQue* evaluation framework [52]. All surveys are presented in Appendix B with their questions and an overview on the results.

6.3.3. Variant B: Social Matching

Before we continue with the participant, we must first attach groups to some events. In variant A, the participant only stated his interests vocally, so the 10 concerts in the red "recommendations" area did not change. We select five of these concerts (specifically those in positions 2, 4, 6, 8, and 10) and attach the groups presented in Table 6.5, mapping concerts 1–5 to events 1–5. For this mapping, we manually search the database to retrieve the primary keys of the five concerts and we use the `GroupDataCreatorExtendingLiveDB.java` program to store the groups and profile data into the database. As a result, the recommendation values of the five enhanced concerts increase and the order of the ten total concerts very likely changes. However, the ten concerts of variant B are still the same as in variant A.

Afterwards, we explain to the participant that we extended the event recommender with social matching features: If he wanted to go to a concert, but could not find companions, then he could tap the "Create group"-button to find other users to join him in his visit. However, we then explain that we would not use this function to create groups in this study, but we imagine that other users have done so previously.

The task is then the same as in variant A: the participant must view the event details screens of the 10 concerts in the red "recommendations" area, thereby also seeing whether groups exist for a concert. He must then vocally state whether he is interested, but now the groups and group members may influence his decision. At the end, he must again rank the top 3 of these 10 concerts. Finally, he receives a survey consisting of 11 questions (9 of which are the same as in variant A), also modelled after the *ResQue* evaluation framework [52].

6.3.4. Conclusion

After both scenarios are complete, the participant receives a demographics survey. We then discuss the course of events, usually leading into the discussion by asking the participant to guess what the goal of this evaluation was. This discussion flows freely and participants can voice their thoughts on the app, the social event extension, and the study itself. At the end, we reveal the motivations of this study and thank the participant for their time.

After each iteration, we reset the starting conditions: we reset the database by manually deleting all groups, and we reset the user's history (i.e. his likes and dislikes for events) via the "Delete history"-button in the app's settings.

6.4. Results

In this section, we present the results of this study. We begin with an overview on the study participants and then present the results to each research question. These results are then discussed in Section 6.5.

6.4.1. Study Participants

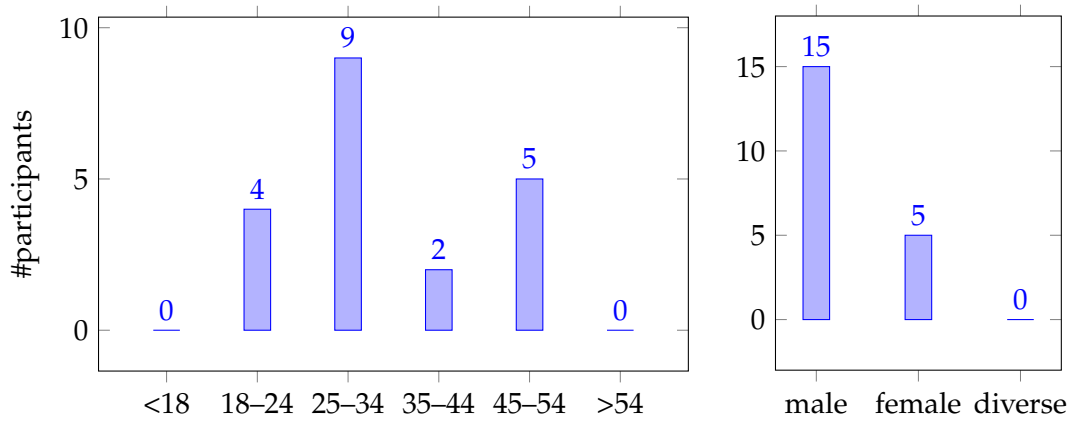


Figure 6.2.: *Demographic distribution of study participants*

The sample of study participants should ideally be a random sample from the target group of potential users [31]. For this purpose, we recruited 20 testers and their demographic distribution is presented in Figure 6.2.

In the age distribution, we focus on two aspects: First, we include people from age groups that currently use the app. In Section 4.2, we analyzed the demographics of

the existing user base and Table 4.1a showed that 64.5% of users are age 35 and above. To cover this demographic, we recruited 7 testers. Second, we also include potential future users by recruiting 13 study participants from younger age ranges.

Table 4.1a has also shown a gender ratio of 66% male to 34% female. We recruited 15 males and 5 females, so the gender ratio of our study sample is similar.

Study participants rated themselves as very experienced in the usage of smartphones and apps (μ: 4.55, σ: 0.75). They occasionally visit paid events, such as concerts, plays, and exhibitions (μ: 3.08, σ: 1.05). If they wanted to go to an event, the likelihood of finding a companion varied greatly – but no one stated that they were very likely to not find one (μ: 2.75, σ: 1.09). Most people visit events with other people (μ: 4.30, σ: 0.71) and almost everyone (95%) finds events more interesting when they can visit with other people (μ: 4.40, σ: 0.54). If participants are interested in the people they're with, then they prefer to rate that "the event they're at does not matter much" (μ: 3.30, σ: 1.01). Consequently, participants tend to disagree with the statement: "If I'm interested in the event I'm at, the people I'm with do not matter much" (μ: 2.50, σ: 1.25).

6.4.2. RQ1: Are users more interested in events with social matching?

First of all, we survey participants whether they found that "The events recommended to me match my interests". Testers tended to agree with this statement (μ_A: 3.65, σ_A: 0.75). However, the social matching functions had no significant impact (μ_B: 3.65, σ_B: 0.67, p: 0.89). We could test this impact for significance using the *Mann-Whitney U-Test*, but this is pointless as $\mu_A = \mu_B$.

We then evaluate the amount of likes the testers have given. In each scenario, each of the 20 testers rated ten events, so the maximum total amount of likes was 200. The quality of the recommended events varies greatly: some testers liked nine of the ten events, others only one in ten. The total amount of likes was the same for both variants A and B (94 likes). Five participants liked more events in variant B, due to the influence of joining groups; however, this is mitigated by five other participants, who liked less events in variant B, partly due to having thoughts about not fitting into the crowd.

To sum it up, our study cannot confirm that RQ1 is true. We found no significant change in a user's interest in events between the original event recommender and the version that was extended with social matching features.

6.4.3. RQ2: Do recommendations improve with social matching?

We answer this question from a multitude of viewpoints using eight metrics: four survey questions rated by the participant, and four derived from his preferences. We begin with the four survey questions (also visualized in Figure 6.3):

First, we analyze whether recommended events match the user's interest better with social matching. This metric was already used in RQ1 and we found that there was no difference (p: 0.89).

Second, we evaluate whether participants agree that "The events recommended to me are diverse". Users were neutral in this question (μ_A: 3.25, σ_A: 1.02), and this did not significantly improve with social matching (μ_B: 3.35, σ_B: 0.99, p: 0.67).

Third, participants agreed with "Overall, I am satisfied with the recommender" (μ_A: 3.70, σ_A: 0.73), this did not significantly improve in variant B (μ_B: 3.75, σ_B: 0.64, p: 0.94).

Fourth, participants also agreed with "I would use this recommender again" (μ_A: 3.95, σ_A: 1.00), but social matching provided no significant improvement (μ_B: 4.05, σ_B: 0.95, p: 0.81).

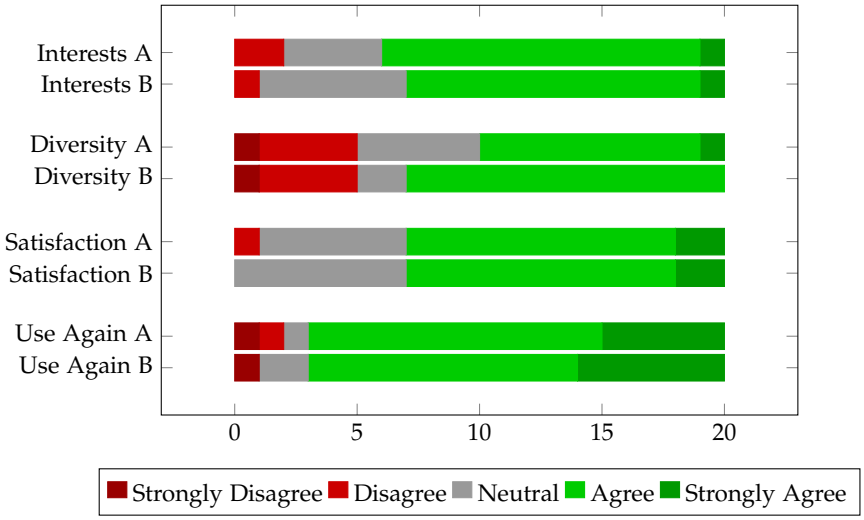


Figure 6.3.: *RQ2: Survey metrics*

We then evaluate four metrics gathered during the scenario:

First, we analyze the positions of the first liked events. The recommender should feature events that the user likes and they should be presented as early as possible. Figure 6.4 shows that most people ($\geq$ 80%) liked the first or second event; this is pleasing as these two events are also directly shown on the start page without scrolling. However, when we compare variants A and B (μ_A: 1.55, σ_A: 1.05, and μ_B: 1.90, σ_B: 1.29, p: 0.31), there are no statistically significant differences to be found.

Second, we know from RQ1 that the same amount of events are liked in both variants. Therefore, we would prefer if variant B changed the distribution so that more events are liked in higher positions. Figure 6.5 presents positions of all liked events; there is no significant difference (μ_A: 5.10, σ_A: 2.92, and μ_B: 5.20, σ_B: 2.91, p: 0.81).

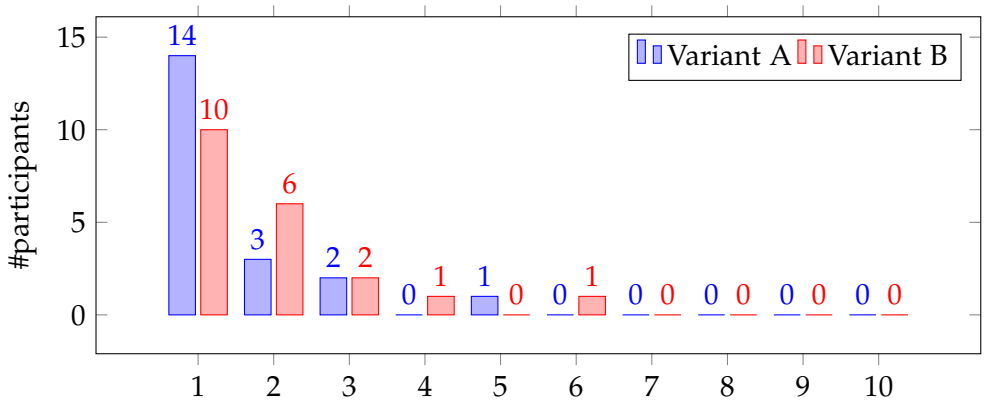


Figure 6.4.: *Positions of first liked events (n=20)*

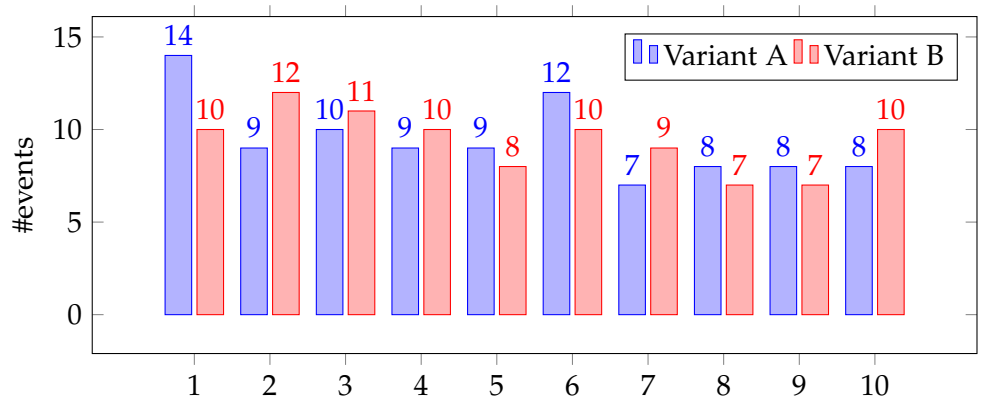


Figure 6.5.: *Positions of all liked events (n=94 in both variants)*

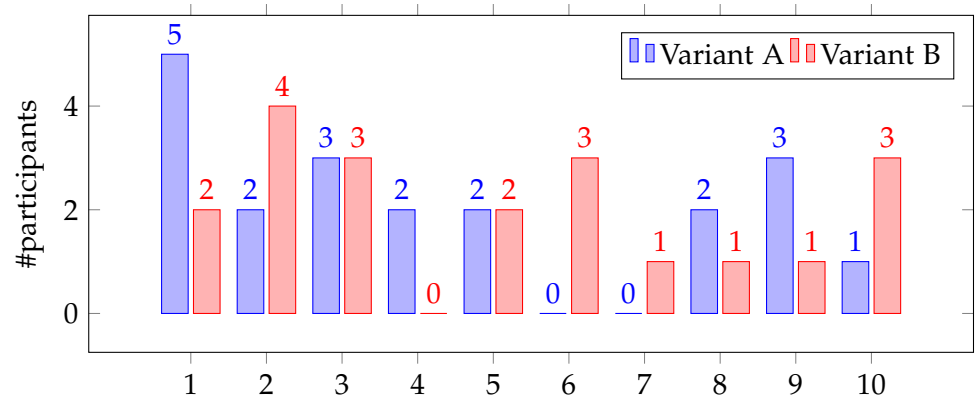


Figure 6.6.: *Positions of most liked events (n=20)*

Third, we evaluate the positions of the most liked events: a high position, ideally the first, suggests that the quality of the recommender is high. Figure 6.6 visualizes the participant's responses in variants A (μ_A: 4.45, σ_A: 3.22) and B (μ_B: 5.05, σ_B: 3.14, p: 0.47). Again, the data supports no significant changes between the two variants.

Fourth, we evaluate the positions of the top3 liked events; a higher position is better. In both variants, the mean position of the top 3 liked events is near the mean of the entire range of 1–10 (μ_A: 5.08, σ_A: 2.95, and μ_B: 5.15, σ_B: 3.09, p: 0.94), and we cannot find a significant change.

In summary, we cannot confirm that RQ2 is true, since we could not find significant differences in any of the eight metrics of RQ2.

6.4.4. RQ3: Do social matching features negatively impact the usability?

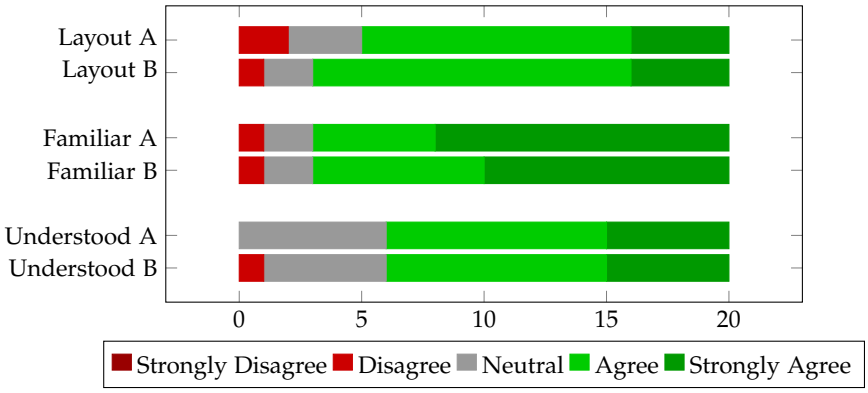


Figure 6.7.: *RQ3: Survey metrics*

To answer this question, we evaluate three survey metrics (visualized in Figure 6.7):

First, we analyze whether participants found "the layout and labels of the recommender interface adequate". Testers did agree with this statement in variant A (μ_A: 3.85, σ_A: 0.88); the same is true for variant B (μ_B: 4.00, σ_B: 0.73, p: 0.67). There are no statistically significant changes.

Second, testers strongly agreed in both variants that they "became familiar with the recommender system quickly"; only one person disagreed. Again, we find no significant changes (μ_A: 4.40, σ_A: 0.88, and μ_B: 4.30, σ_B: 0.87, p: 0.65).

Third, participants agreed that they "understood why events were recommended to them". We cannot find any significant differences for this question between variants A and B (μ_A: 3.95, σ_A: 0.76, and μ_B: 3.90, σ_B: 0.85, p: 0.94).

To sum it up, we conclude that our social matching features do not negatively impact the usability, since there are no significant changes in the metrics telling us otherwise.

6.4.5. RQ4: What influences do demographic factors have?

In this research question, we evaluate the same nine metrics as in RQ1 and RQ2, but we split our 20 participants into two groups. Our reasoning for this research question was that one could assume that young students new to the city are more enthusiastic to use social matching features than older people with established friend groups and family. This is why we first evaluate differences in age groups, and then follow up with other demographic factors, such as gender and personality.

Differences in Age Groups

We split our 20 participants into two age groups as previously defined: 7 testers of age 35 and above, representing the existing user base; and 13 testers of age 18–34, representing potential future users. We then only compare the answers of these two groups for variant B, this simulates a between-subjects A/B test where age is the difference.

First, the older age group was neutral on whether the events matched their interests (μ_{35+}: 3.29, σ_{35+}: 0.49). The younger age group, however, agreed that events matched their interests ($\mu_{18–34}$: 3.85, $\sigma_{18–34}$: 0.69). Using the *Mann-Whitney U-Test* we find that this difference is not statistically significant ($p = 0.06$).

Second, the older age group liked less events per person compared to the younger age group, however this difference was not significant (μ_{35+}: 4.00, σ_{35+}: 2.00, and $\mu_{18–34}$: 5.07, $\sigma_{18–34}$: 2.29, p: 0.28).

Third, the older age group somewhat agreed that events were diverse (μ_{35+}: 3.57, σ_{35+}: 1.13); the younger age group were neutral in this question ($\mu_{18–34}$: 3.08, $\sigma_{18–34}$: 0.95). There was no significant difference (p: 0.30).

Fourth, both younger and older age groups agreed that they were satisfied with the recommender overall; there were no significant differences (μ_{35+}: 3.86, σ_{35+}: 0.69, and $\mu_{18–34}$: 3.62, $\sigma_{18–34}$: 0.77, p: 0.60).

Fifth, both groups agreed that they would use the recommender again; there were no significant differences (μ_{35+}: 4.27, σ_{35+}: 0.49, and $\mu_{18–34}$: 3.77, $\sigma_{18–34}$: 1.17, p: 0.45).

Sixth, there were no significant differences in the positions of the first liked events (μ_{35+}: 2.57, σ_{35+}: 1.90, and $\mu_{18–34}$: 1.54, $\sigma_{18–34}$: 0.66, p: 0.32)

Seventh, we did not find significant differences in the positions of all liked events (μ_{35+}: 5.11, σ_{35+}: 2.82, and $\mu_{18–34}$: 5.24, $\sigma_{18–34}$: 2.97, p: 0.88)

Eighth, no significant differences were found in the positions of the most liked events (μ_{35+}: 5.86, σ_{35+}: 3.80, and $\mu_{18–34}$: 3.69, $\sigma_{18–34}$: 2.72, p: 0.32).

Ninth, we also did not find significant differences between the positions of the top 3 liked events (μ_{35+}: 5.81, σ_{35+}: 3.01, and $\mu_{18–34}$: 4.69, $\sigma_{18–34}$: 2.88, p: 0.18).

Differences in Other Demographic Factors

We found $p = 0.06$ between younger and older people in the question ("The events recommended to me match my interests"), so we evaluate this question further, looking at other demographic factors:

First, the 15 men and 5 women like events the same, there are no significant differences (μ_{male}: 3.60, σ_{male}: 0.74, and μ_{female}: 3.80, σ_{female}: 0.45, p: 0.63).

Second, 7 participants stated that they go to events "frequently" (or "very frequently") and 13 testers go to events only "occasionally" or less. There are no significant differences between these two groups in whether events matched their interests ($\mu_{freq.}$: 3.43, $\sigma_{freq.}$: 0.98, and $\mu_{occ.}$: 3.77, $\sigma_{occ.}$: 0.44, p: 0.36).

Third, 5 participants stated that it was likely that "want to go to an event but can't find a companion", 15 testers did not. Again, there are no significant differences between these two groups in whether events matched their interests (μ_{likely}: 3.80, σ_{likely}: 0.45, and $\mu_{notlikely}$: 3.60, $\sigma_{notlikely}$: 0.74, p: 0.63).

In summary of RQ4, we did not find significant differences in the metrics.

6.5. Discussion

In this section, we discuss the results presented in Section 6.4. After each study, we discussed the course of events with the participant. They often explained their behaviour, and voiced their thoughts on the app and the social matching extension. We first group these free-text answers and then discuss each research question individually.

Selection of Recommended Events

Some participants had difficulties in deciding whether to like or dislike a concert.

It was, for example, very difficult to do so when an event had no description and only a default image, consisting of just a name, genre, and location. In other cases, participants received similar "generic" events: one tester was presented with three jazz-nights in bars, another received two jam sessions of the blues. This "bubble" of similar events was also true for the selection of concerts based on the genre. If a participant had, for example, liked two jazz concerts in the first batch of ten events during the setup, then the following batches were very likely to only show jazz concerts thereafter, since we tweaked the algorithm to sort events based on their recommendation value and disabled other diversity functions.

Many participants were not sure whether to be interested in an artist they did not know. Liking an event then became a game of "hunting for an artist [they] knew".

Selection of Other People

A few testers criticized that the fictitious users displayed in the groups were unrealistic and that there was not enough information to form an opinion on them. Some testers proposed that a user should be able to enter free-text information in his profile (such as his hobbies) for others to see. One participant mentioned that they could not imagine themselves meeting other people via our implementation ("serial killers!"), due to low trust and that we would have to implement major verification and rating features.

Usage of Social Matching

Participants rarely looked at the individual people inside the groups. For most people, the event characteristics were much more important than potential groups, some testers even ignored the social matching features entirely. One tester proposed an improvement: He declared that he would be more interested in social matching features if the algorithm explained that they were the reason for why an event was recommended.

RQ1: Are users more interested in events with social matching?

The metrics of RQ1 did not show a significant improvement in a user's interest for events when social matching features were activated. A large number of testers explained that they already had established friend groups (and family) whom they would rather go to events with. Some testers ignored the social matching features entirely and were not interested in getting to know new people, as mentioned in the previous paragraph.

Others embraced the social matching features, stating that their interest for an event was increased when they saw that it was populated with groups, since this signified that the event was popular. Some testers were more interested in an event because of the groups; but on the other hand, other testers disliked an event more when they thought that they would not fit into the crowd ("the event is not for me").

The study has shown that almost everyone finds events more interesting when they can visit with other people and that it is important who these people are. When asked, participants agree that the idea of social matching is sound and that it could make events more interesting; however, the participant's actions in the study tell us that they mostly think so for "other people". No tester was actively looking for new friends, and this was in a study setting – the barrier to an actual meet-up of strangers would be much higher in practice.

RQ2: Do recommendations improve with social matching?

The metrics of RQ2 did not show a significant improvement of recommendations with social matching. A few participants explained that it would be more interesting to find the friends first and to then decide which event to go to. They also told us that the study's limitation to concerts was not helpful, since visitors would follow the performance and not talk to one another ("parties may be better").

In general, we find that the social matching algorithm's impact on an event's recommendation value is too strong. The presentation of the ten events in variant B almost always ranked the five events with groups at positions 1–5. The metrics show that variant B might actually be a little worse (but not significantly), e.g. in Figure 6.6 with the position of the most liked event: $\mu_A = 4.45$ vs. $\mu_B = 5.05$.

We overvalued the impact of social matching to the general target user base. It might be of value to a smaller subset of users, this will be discussed in RQ4. One solution to this problem is to tweak the algorithm's static values, another approach could be to separate the social matching algorithm from the event recommender algorithm.

RQ3: Do social matching features negatively impact the usability?

The metrics of RQ3 show that variants A and B have no significant differences in the experience of the user interface. Study participants were very experienced in the usage of smartphones and apps (μ: 4.55) and they understood how to use the app easily. Some testers proposed improvements to the pre-existing app, such as larger fonts, but no one had remarks on the social matching user interfaces.

We conclude that our implementation of the social matching user interface did not negatively impact the usability of the app. It was therefore not detrimental to the other research questions.

RQ4: What influences do demographic factors have?

The metrics of RQ4 show no significant differences due to demographic factors. However, there could be differences between younger and older people: The survey response for the question ("The events recommended to me match my interests") differed with a p-value of $p = 0.06$, and future work should investigate this further.

These results raise questions and possibilities for the future of the app: Which subsets of people do find social matching features in event recommenders interesting and what is the ratio of these people to the total potential user base?

Summary

In summary, we find that the study participants did not perceive our social matching implementation to be of significant value. We explained the reasons for this above, for example that participants were simply not that interested in getting to know "unrealistic" fictitious people. Future work could then mitigate these factors, to be explained in Section 7.

6.6. Threats to Validity

In this section, we describe the threats to the validity of our study.

Study Design

This study used a sequential within-subjects study design [31] to perform A/B testing. Each tester experienced both variants A and B, leading to a carryover effect, meaning that the second scenario might be biased by the first. To mitigate this, we alternated the order of variants A and B for each tester.

Another threat is that we conducted a field study, compared to an alternative lab study on the database. Participants might have been biased by the circumstances of the evaluation.

Study Objects

We limited the app to not show all kinds of events, but only concerts; this was done for performance reasons. We assumed that this change would produce only negligible differences in the behaviour of our study participants. However, in the evaluation some participants stated that the study's limitation to concerts was not helpful, since concerts are not that social compared to other events (such as parties).

We also presented study participants with fictitious users for social matching, since it was logistically unfeasible to match study participants to one another. Participants complained that these fictitious users were unrealistic. We did mitigate this threat by creating a diverse set of fictitious users with realistic names, as described in Section 6.2, but we did not focus on the selection of "normal" profile pictures and instead chose attractive ones.

Study Participants

Participants might not be potential users of the social matching features, even if they are in the target group that would use the pre-existing event recommender. We mitigated

this by recruiting participants from a target group that is diverse in age, gender, and personality.

We recruited participants from Munich in Germany. It is possible that German culture is more reserved towards making new friends. We tried to mitigate this threat by including people with migration backgrounds.

The most important threat is that the recruitment of participants was limited only by whether participants have been to concerts in the last 2 years and whether they are experienced with smartphones and app. If people who have no desire to get to know other people in the first place, then social matching provides cannot provide value. It is valid that we included such people in the evaluation, since they are part of the current user base, and since we want to know what effect social matching has over the entire range of users. However, future work should evaluate social matching for the subset of users that are interested in social matching to then quantify how much value our implementation can provide.

Study Procedure

We might have influenced participants during the procedure of the study. When we, for example, explained that we would need a top 3 ranking of events, participants might be influenced to like at least 3 events. We did mitigate many influential factors, such as separating participants in location and following the same script for each participant.

Sample Size

Finally, the sample size of 20 should be increased. Knijnenburg et al. describe that 38 participants are needed to detect medium-sized effects (0.3 standard deviations at a power level of 85%) [31]. Thus, we can only detect large-sized effects with significance; for medium-sized (and smaller) effects, we can only presume trends, e.g. that variant B might be worse in the ranking of the most liked event, as presented in Figure 6.6.

7. Future Work

This chapter presents ideas for future work which were beyond the scope of this thesis. We begin with the implementation and then follow up with the evaluation.

Expanding the Implementation

In this thesis, we implemented a minimum viable system that allows users to join groups for events. This system was defined by user stories 4–9 (see Section 4.4), and we also included user stories 1–3. We were able to use this prototype implementation in the evaluation; if social matching features are to be brought into production however, one should consider the following points:

First, missing user stories should be implemented, such as notifications, group filtering mechanisms, or user rating features. These features would enhance the usability of the system greatly.

Second, one should research ways for users to contact each other. Currently, we display an email address and users need to contact each other manually. This is not very user-friendly, and group chats or forums could improve this. In general, one should consider how many social networking functions should be included – the system might evolve from an event recommender into an event-based social network.

Third, the user profile should be enhanced. Currently, users can upload a profile picture and show their name and age to other users. This is not enough information for a user to create a meaningful opinion on the other users. To improve this, one could allow multiple profile pictures, so that users can present themselves better. One could also introduce profile text boxes, so that users can state their interests and hobbies.

Fourth, we only implemented the social matching features for the Apple iOS app; one should also implement these features on Android. The server does not need to be modified.

Fifth, one should research which other use cases social matching can be applied to. Currently, we allow users to create groups for any event in the event recommender. In practice, this might lead to many groups of one person only, since other users might not be interested in small obscure events. One could, for example, allow groups only on featured events from more well-known artists.

Improving the Evaluation

In Section 6.6, we presented the threats to the validity of the evaluation. These threats can be used as basis for future work:

First, the test can be expanded to include more participants, possibly using a between-subjects study design instead of the within-subjects design we used. This would increase the power of the evaluation, and when using a within-subjects design, we would eliminate the carry-over effect.

Second, the evaluation can also be conducted the with various selections of events. Our evaluation used a large amount of concerts, consequently, users might be less interested in these concerts when they don't recognize the artist. One could, for example, limit the events to only feature popular artists. One could also remove the limitation to only concerts (since users stated that they found concerts not very social) or research other variants, for example to only show parties.

Third, various thresholds and values for the social matching algorithm need to be evaluated. We explained in Section 5.3 that we selected the values subjectively after a manual pre-trial. We found that we overstated the value of social matching for the general target population, so the algorithmic values need to be tweaked. One could also reduce the integration between the social matching algorithms into the event recommendation algorithms.

Fourth, different kinds of evaluations should be performed. This includes lab studies on the existing database, this would remove individual biased of participants in the evaluation. One could also design a field study in which real study participants are matched to each other, for example in an alpha-test of the implementation. This would mitigate the use of fictitious people which testers found unrealistic. Finally and most importantly, different subgroups of the target population should be evaluated, for example (young) people who are actively looking for friends. Social matching could be of significant importance for these subgroups, and future work should evaluate this.

Other Ideas

In the following, we present other ideas for future research:

First, one could research how different user interfaces influence the user's behaviour. In our implementation, we were limited in the freedom of our design, because our extension of the system was an optional feature and had to fit in with the pre-existing interface. An integrated system, designed from the group up for both social matching and event recommendations, could possibly be more engaging for a user. One could, for example, investigate how prominently other users can be featured, how one should place profile pictures and in what size.

Second, one could investigate other ideas for how social matching features could be integrated into event recommenders. Our system attaches groups to events, thus users must first find an interesting event and can then see the groups. But one could also allow users to find other interesting users first; these groups could then be recommended events using group recommendation techniques. In general, one needs to think about use cases for the intersection of recommending events and recommending people: users who are mainly interested in getting to know other people could possibly prefer people-to-people recommenders more than event recommenders.

8. Conclusion

This thesis contributes three main results to the field of recommender systems, specifically for event recommenders and reciprocal recommenders:

First, we elicitated requirements for an event recommender that includes reciprocal social matching functions in Chapter 4. We were to extend a pre-existing event recommender app; therefore, we first analyzed its current user base. We then examined existing approaches to social matching and event recommendation. Finally, we presented functional requirements in the form of user stories from the users' perspective; we also created wireframes that modelled the user interface. These resulting artifacts, i.e. user stories and wireframes, can be used as a basis for the development of recommenders in other scenarios.

Second, we implemented a prototype that enables users to join groups for events in Chapter 5. We also described how we expanded the app how the server was enhanced to support this app. Finally, we extended the pre-existing event recommender algorithm to integrate a social matching algorithm that consists of three stages: considering demographic information, finding similar users, and computing reciprocal matches. The descriptions of app and server can be used for comparison with other implementations, and the reciprocal algorithm can be used as a basis for future discussion, adaptation, and improvement.

Third, we evaluated our implementation in Chapter 6. We found that RQ1 ("Are users more interested in events with social matching?") is not true and that RQ2 ("Do recommendations improve with social matching?") also does not hold. RQ3 ("Do social matching features negatively impact the suability?" is also negated. In summary, we found that study participants did not perceive our social matching implementation to be of significant value. The main reason for this is that participants were not actively interested in getting to know other people. In RQ4 ("What influences do demographic factors have?") we also did not find significant differences, but we were able to raise further research questions: One should study how much value social matching can have in event recommenders for people who are actively interested in getting to know other people. One could also differentiate further between demographic niches.

List of Figures

List of Listings

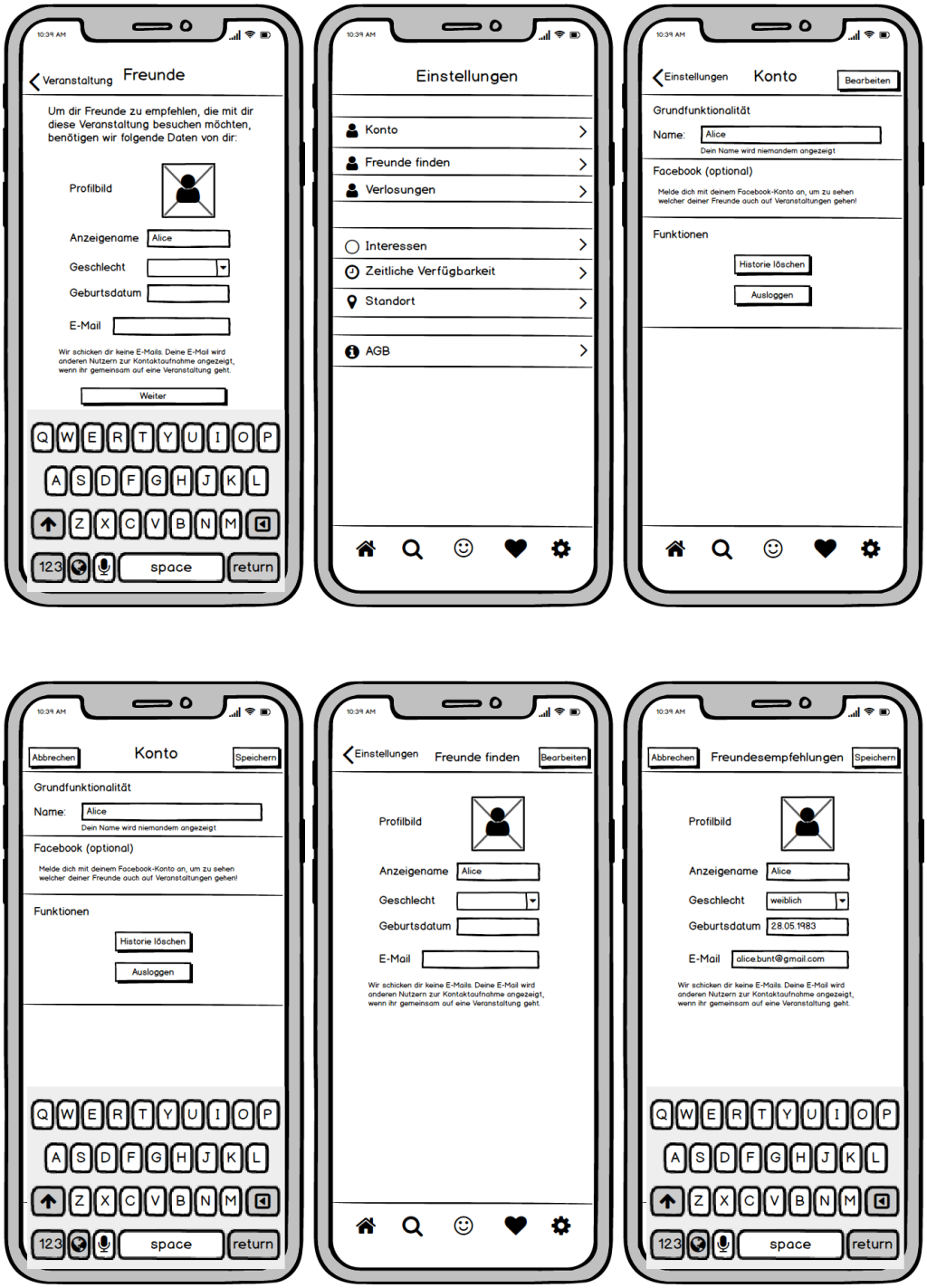
List of Tables

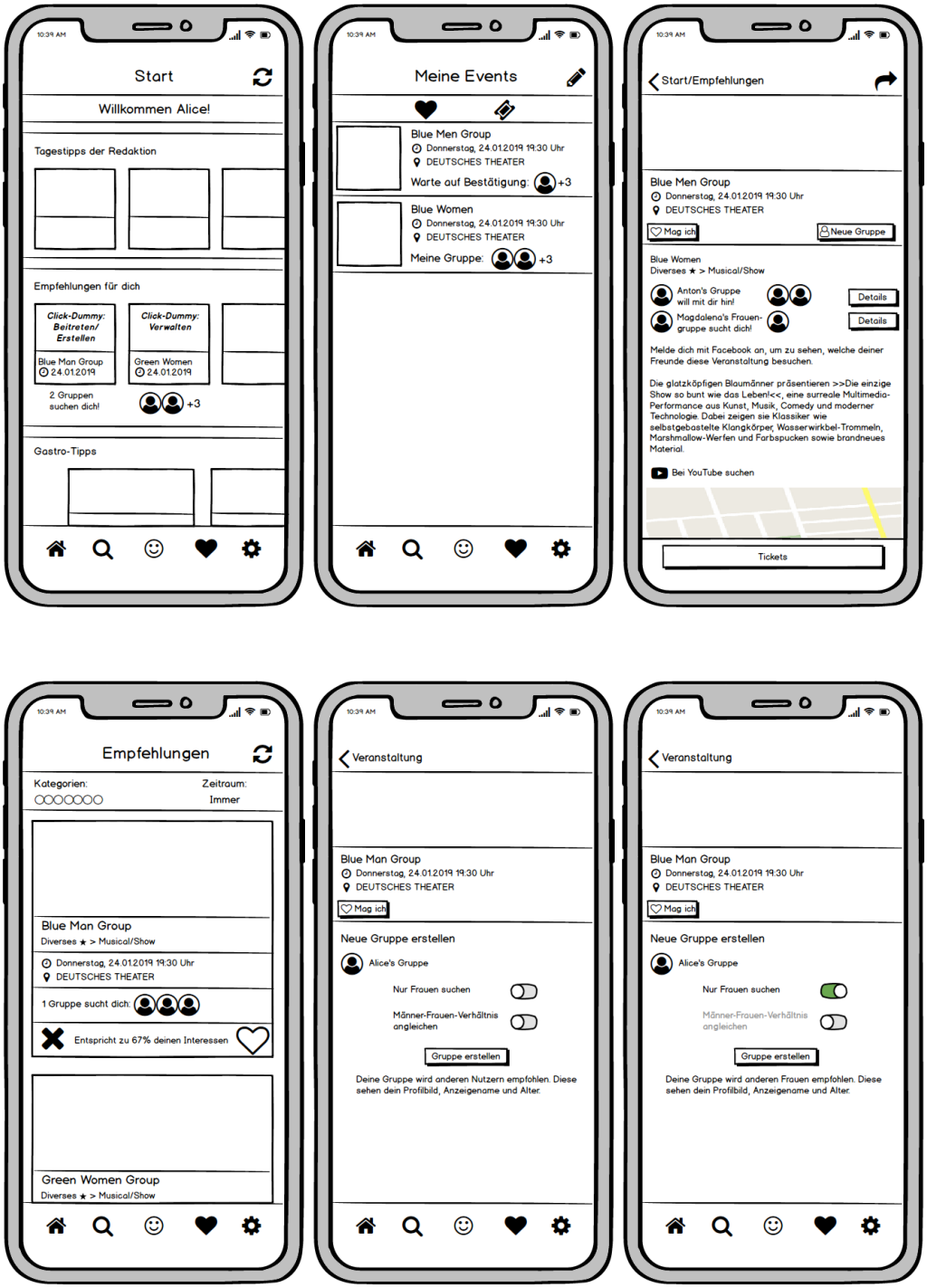
List of Abbreviations

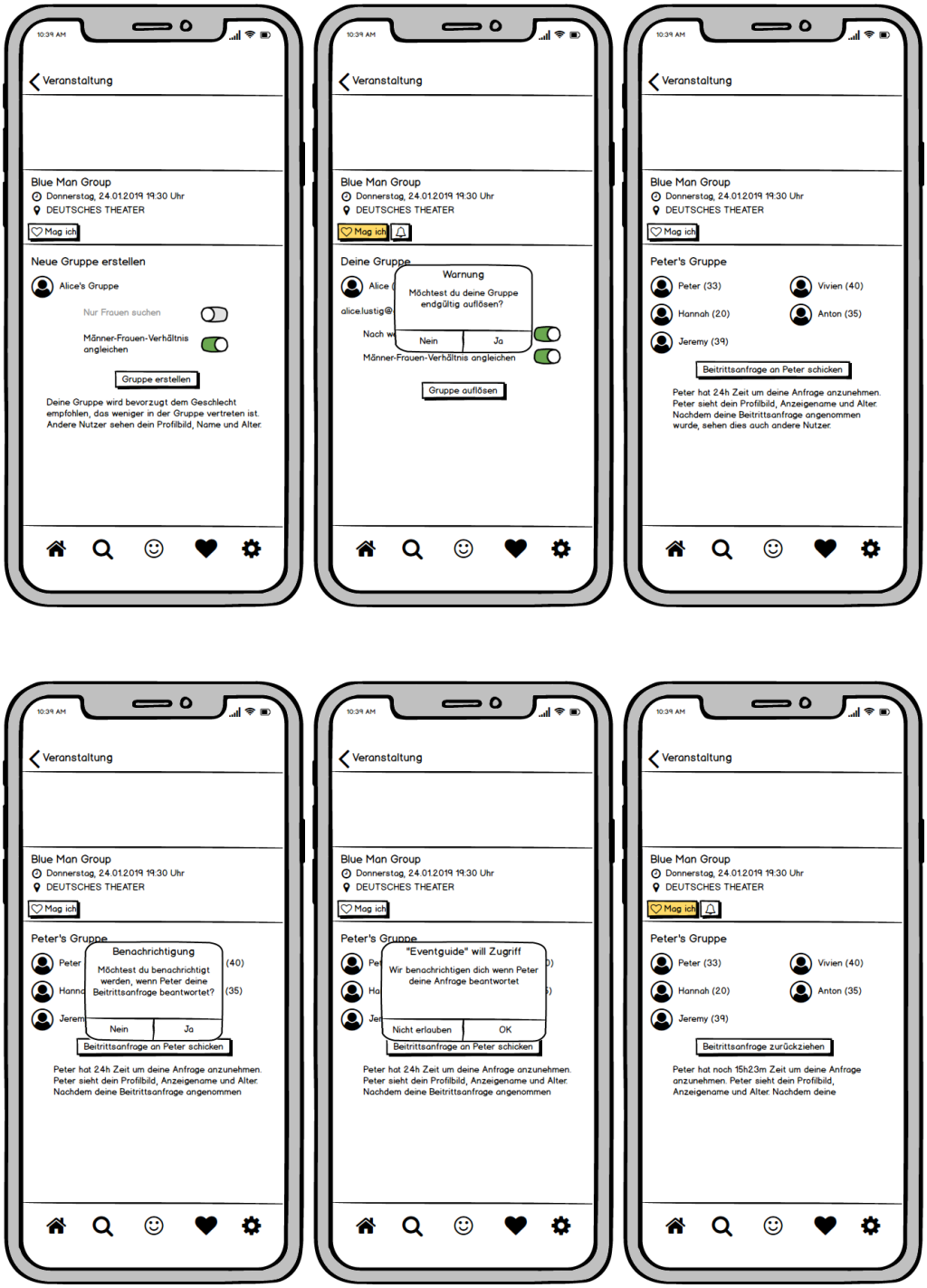
API	Application Programming Interface
CBCF	Content-Boosted Collaborative Filtering
DAO	Data Access Object
GQM	Goal-Question-Metric
JSON	JavaScript Object Notation
RDBMS	Relational Database Management System
RECON	Reciprocal Recommender for Online Dating
REST	Representational State Transfer
SQL	Structured Query Language

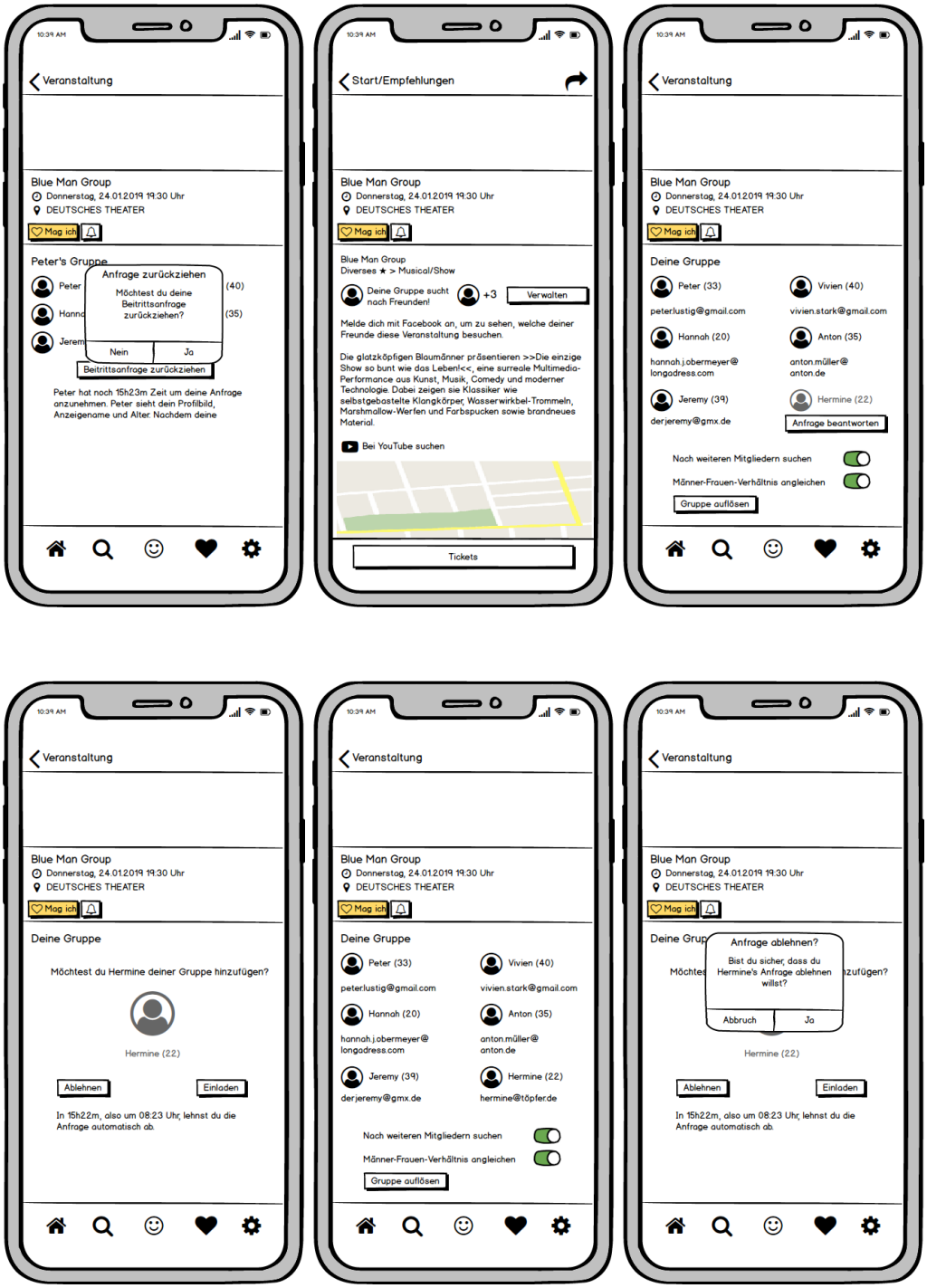
A. Interactive prototype

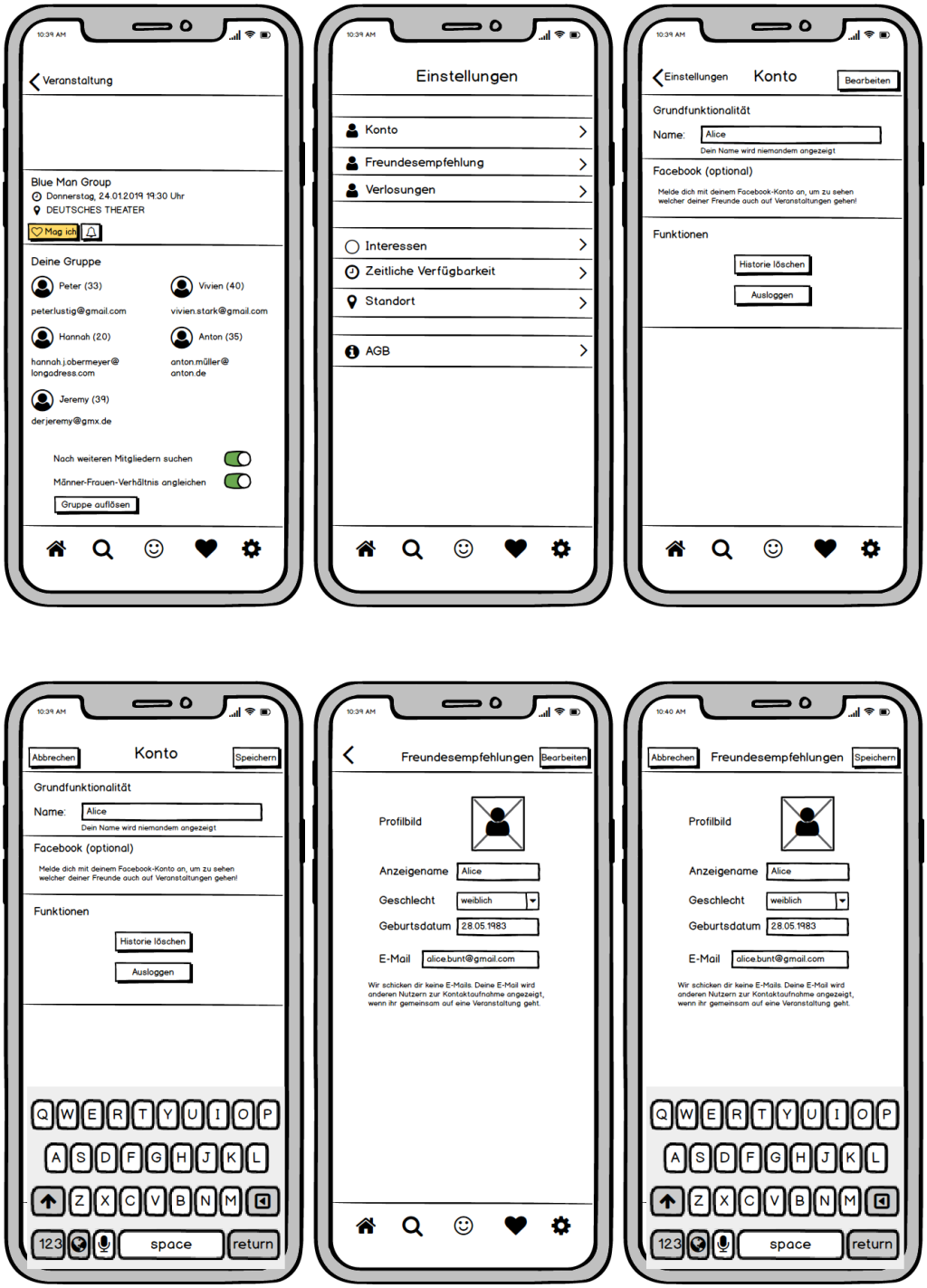
In the following, we present 33 screens from the interactive prototype:


Veranstaltung
Freunde
Um dir Freunde zu empfehlen, die mit dir diese Veranstaltung besuchen möchten, benötigen wir folgende Daten von dir:
Profilbild
Anzeigename
Alice
Geschlecht
Geburtsdatum
E-Mail
Wir schicken dir keine E-Mails. Deine E-Mail wird anderen Nutzern zur Kontaktaufnahme angezeigt, wenn ihr gemeinsam auf eine Veranstaltung geht.
Weiter
Einstellungen
Konto
Freunde finden
Verlosungen
Interessen
Zeitliche Verfügbarkeit
Standort
AGB
Einstellungen
Konto
Bearbeiten
Grundfunktionalität
Name:
Alice
Dein Name wird niemandem angezeigt
Facebook (optional)
Melde dich mit deinem Facebook-Konto an, um zu sehen welcher deiner Freunde auch auf Veranstaltungen gehen!
Funktionen
Historie löschen
Ausloggen
Abbrechen
Konto
Speichern
Grundfunktionalität
Name:
Alice
Dein Name wird niemandem angezeigt
Facebook (optional)
Melde dich mit deinem Facebook-Konto an, um zu sehen welcher deiner Freunde auch auf Veranstaltungen gehen!
Funktionen
Historie löschen
Ausloggen
Einstellungen
Freunde finden
Bearbeiten
Profilbild
Anzeigename
Alice
Geschlecht
Geburtsdatum
E-Mail
Wir schicken dir keine E-Mails. Deine E-Mail wird anderen Nutzern zur Kontaktaufnahme angezeigt, wenn ihr gemeinsam auf eine Veranstaltung geht.
Abbrechen
Freundesempfehlungen
Speichern
Profilbild
Anzeigename
Alice
Geschlecht
weiblich
Geburtsdatum
28.05.1983
E-Mail
alice.bunt@gmail.com
Wir schicken dir keine E-Mails. Deine E-Mail wird anderen Nutzern zur Kontaktaufnahme angezeigt, wenn ihr gemeinsam auf eine Veranstaltung geht.
space
return


Start
Willkommen Alice!
Tagestipps der Redaktion
Empfehlungen für dich
Click-Dummy: Beitreten/ Erstellen
Blue Man Group
24.01.2019
Click-Dummy: Verwalten
Green Women
24.01.2019
2 Gruppen suchen dich!
+3
Gastro-Tipps
Meine Events
Blue Men Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Warte auf Bestätigung:
+3
Blue Women
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Meine Gruppe:
+3
Start/Empfehlungen
Blue Men Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Mag ich
Neue Gruppe
Blue Women
Diverses ★ > Musical/Show
Anton's Gruppe will mit dir hin!
Details
Magdalena's Frauen-gruppe sucht dich!
Details
Melde dich mit Facebook an, um zu sehen, welche deiner Freunde diese Veranstaltung besuchen.
Die glatzköpfigen Blaumänner präsentieren >>Die einzige Show so bunt wie das Leben!<<, eine surreale Multimedia-Performance aus Kunst, Musik, Comedy und moderner Technologie. Dabei zeigen sie Klassiker wie selbstgebastelte Klangkörper, Wasserwirkbel-Trommeln, Marshmallow-Werfen und Farbspucken sowie brandneues Material.
Bei YouTube suchen
Tickets
Empfehlungen
Kategorien:
Zeitraum:
Immer
Blue Man Group
Diverses ★ > Musical/Show
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
1 Gruppe sucht dich:
Entspricht zu 67% deinen Interessen
Green Women Group
Diverses ★ > Musical/Show
Veranstaltung
Blue Man Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Mag ich
Neue Gruppe erstellen
Alice's Gruppe
Nur Frauen suchen
Männer-Frauen-Verhältnis angleichen
Gruppe erstellen
Deine Gruppe wird anderen Nutzern empfohlen. Diese sehen dein Profilbild, Anzeigename und Alter.
Veranstaltung
Blue Man Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Mag ich
Neue Gruppe erstellen
Alice's Gruppe
Nur Frauen suchen
Männer-Frauen-Verhältnis angleichen
Gruppe erstellen
Deine Gruppe wird anderen Frauen empfohlen. Diese sehen dein Profilbild, Anzeigename und Alter.


Veranstaltung
Blue Man Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Mag ich
Neue Gruppe erstellen
Alice's Gruppe
Nur Frauen suchen
Männer-Frauen-Verhältnis angleichen
Gruppe erstellen
Deine Gruppe wird bevorzugt dem Geschlecht empfohlen, das weniger in der Gruppe vertreten ist. Andere Nutzer sehen dein Profilbild, Name und Alter.
Deine Gruppe
Warnung
Möchtest du deine Gruppe endgültig auflösen?
Nein
Ja
Männer-Frauen-Verhältnis angleichen
Gruppe auflösen
Peter's Gruppe
Peter (33)
Vivien (40)
Hannah (20)
Anton (35)
Jeremy (39)
Beitrittsanfrage an Peter schicken
Peter hat 24h Zeit um deine Anfrage anzunehmen. Peter sieht dein Profilbild, Anzeigename und Alter. Nachdem deine Beitrittsanfrage angenommen wurde, sehen dies auch andere Nutzer.
Benachrichtigung
Möchtest du benachrichtigt werden, wenn Peter deine Beitrittsanfrage beantwortet?
Nein
Ja
"Eventguide" will Zugriff
Wir benachrichtigen dich wenn Peter deine Anfrage beantwortet
Nicht erlauben
OK
Beitrittsanfrage zurückziehen
Peter hat noch 15h23m Zeit um deine Anfrage anzunehmen. Peter sieht dein Profilbild, Anzeigename und Alter. Nachdem deine


Veranstaltung
Blue Man Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Mag ich
Peter's Gruppe
Anfrage zurückziehen
Möchtest du deine Beitrittsanfrage zurückziehen?
Nein
Ja
Beitrittsanfrage zurückziehen
Peter hat noch 15h23m Zeit um deine Anfrage anzunehmen. Peter sieht dein Profilbild, Anzeigename und Alter. Nachdem deine
Start/Empfehlungen
Blue Man Group
Diverses ★ > Musical/Show
Deine Gruppe sucht nach Freunden!
+3
Verwalten
Melde dich mit Facebook an, um zu sehen, welche deiner Freunde diese Veranstaltung besuchen.
Die glatzköpfigen Blaumänner präsentieren >>Die einzige Show so bunt wie das Leben!<<, eine surreale Multimedia-Performance aus Kunst, Musik, Comedy und moderner Technologie. Dabei zeigen sie Klassiker wie selbstgebastelte Klangkörper, Wasserwirkbel-Trommeln, Marshmallow-Werfen und Farbspucken sowie brandneues Material.
Bei YouTube suchen
Tickets
Deine Gruppe
Peter (33)
peter.lustig@gmail.com
Vivien (40)
vivien.stark@gmail.com
Hannah (20)
hannah.j.obermeyer@longadress.com
Anton (35)
anton.müller@anton.de
Jeremy (39)
derjeremy@gmx.de
Hermine (22)
Anfrage beantworten
Nach weiteren Mitgliedern suchen
Männer-Frauen-Verhältnis angleichen
Gruppe auflösen
Möchtest du Hermine deiner Gruppe hinzufügen?
Hermine (22)
Ablehnen
Einladen
In 15h22m, also um 08:23 Uhr, lehnst du die Anfrage automatisch ab.
Anfrage ablehnen?
Bist du sicher, dass du Hermine's Anfrage ablehnen willst?
Abbruch
Ja


Veranstaltung
Blue Man Group
Donnerstag, 24.01.2019 19:30 Uhr
DEUTSCHES THEATER
Mag ich
Deine Gruppe
Peter (33)
peterlustig@gmail.com
Vivien (40)
vivien.stark@gmail.com
Hannah (20)
hannah.j.obermeyer@longadress.com
Anton (35)
anton.müller@anton.de
Jeremy (39)
derjeremy@gmx.de
Nach weiteren Mitgliedern suchen
Männer-Frauen-Verhältnis angleichen
Gruppe auflösen
Einstellungen
Konto
Freundesempfehlung
Verlosungen
Interessen
Zeitliche Verfügbarkeit
Standort
AGB
Einstellungen
Konto
Bearbeiten
Grundfunktionalität
Name:
Alice
Dein Name wird niemandem angezeigt
Facebook (optional)
Melde dich mit deinem Facebook-Konto an, um zu sehen welcher deiner Freunde auch auf Veranstaltungen gehen!
Funktionen
Historie löschen
Ausloggen
Abbrechen
Konto
Speichern
Grundfunktionalität
Name:
Alice
Dein Name wird niemandem angezeigt
Facebook (optional)
Melde dich mit deinem Facebook-Konto an, um zu sehen welcher deiner Freunde auch auf Veranstaltungen gehen!
Funktionen
Historie löschen
Ausloggen
Freundesempfehlungen
Bearbeiten
Profilbild
Anzeigename
Alice
Geschlecht
weiblich
Geburtsdatum
28.05.1983
E-Mail
alice.bunt@gmail.com
Wir schicken dir keine E-Mails. Deine E-Mail wird anderen Nutzern zur Kontaktaufnahme angezeigt, wenn ihr gemeinsam auf eine Veranstaltung geht.
Abbrechen
Freundesempfehlungen
Speichern
Profilbild
Anzeigename
Alice
Geschlecht
weiblich
Geburtsdatum
28.05.1983
E-Mail
alice.bunt@gmail.com
Wir schicken dir keine E-Mails. Deine E-Mail wird anderen Nutzern zur Kontaktaufnahme angezeigt, wenn ihr gemeinsam auf eine Veranstaltung geht.
space
return

B. Evaluation results

Demographics Survey

We begin with two questions about gender and age:

Survey Question	male	female
"Please choose your gender"	15	5

Survey Question	18–24	25–34	35-44	45-54	55+
"Please select your age group"	4	9	2	5	0

We then ask seven questions to be rated on a 5-point Likert-scale:
In question 1, ratings ranged from "No Experience (1)" to "Advanced (5)".
In question 2, ratings ranged from "Very Rarely (1)" to "Very Frequently (5)".
In questions 3 and 4, ratings ranged from "Very Unlikely (1)" to "Very Likely (5)".
The remaining ratings ranged from "Disagree Strongly (1)" to "Agree Strongly (5)".

Survey Question	1	2	3	4	5
"How would you rate yourself as a smartphone/app user?"	0	1	2	2	15
"How often do you visit paid events (e.g. concerts, plays, exhibitions)?"	2	3	8	6	1
"How likely is it that you want to go to an event but can't find a companion?"	3	5	6	6	0
"If you visit a paid event, how likely are you to be not on your own? (i.e. with others) "	0	1	2	7	10
"Events are more interesting to me when I can visit with other people"	0	1	0	9	10
"If I'm interested in the people I'm with, the event I'm at doesn't matter much"	1	3	7	7	2
"If I'm interested in the event I'm at, the people I'm with don't matter much"	4	7	5	3	1

Survey for Variant A

Ratings ranged from "Disagree Strongly (1)" to "Agree Strongly (5)":

Survey Question	1	2	3	4	5
"The events recommended to me match my interests"	0	2	4	13	1
"The events recommended to me are diverse"	1	4	5	9	1
"The layout and labels of the recommender interface are adequate"	0	2	3	11	4
"I became familiar with the recommender system quickly"	0	1	2	5	12
"I understood why the events were recommended to me"	0	0	6	9	5
"The recommender can be trusted"	0	0	15	4	1
"Overall, I am satisfied with the recommender"	0	1	6	11	2
"I would use this recommender again"	1	1	1	12	5
"I would buy tickets for events, given the opportunity"	0	2	1	8	9

	Positions of Top 3	Positions of Likes
Participant 1	9, 1, 2	1, 9
Participant 2	3, 10, 9	1, 2, 3, 4, 9, 10
Participant 3	8, 5, 6	5, 6, 8
Participant 4	4, 1, 3	1, 2, 4, 7
Participant 5	8, 4, 3	3, 4, 8
Participant 6	5, 6, 8	2, 3, 5, 6, 8
Participant 7	2, 4, 9	2, 4, 5, 7, 9
Participant 8	3, 6, 10	1, 2, 3, 5, 6, 7, 8, 9, 10
Participant 9	1, 4, 7	1, 2, 3, 4, 6, 7, 8, 9, 10
Participant 10	2, 6, 1	1, 2, 4, 5, 6
Participant 11	5, 8, 6	1, 5, 6, 7, 8, 10
Participant 12	1, 6, 5	1, 5, 6
Participant 13	10, 6, 9	2, 3, 4, 6, 7, 8, 9, 10
Participant 14	1, 10, 3	1, 3, 6, 10
Participant 15	9, 1, 5	1, 5, 9
Participant 16	1, 3, 2	1
Participant 17	3, 4, 7	1, 2, 3, 4, 7, 10
Participant 18	4, 8, 10	3, 4, 6, 8, 10
Participant 19	9, 1, 6	1, 3, 6, 9
Participant 20	1, 6, 5	1, 5, 6

Survey for Variant B

Ratings ranged from "Disagree Strongly (1)" to "Agree Strongly (5)":

Survey Question	1	2	3	4	5
"The events recommended to me match my interests"	0	1	6	12	1
"The events recommended to me are diverse"	1	4	2	13	0
"The recommender system helped me discover new people"	2	0	6	7	5
"The information provided for the people is sufficient for me to decide whether to join the group"	2	5	6	5	2
"The layout and labels of the recommender interface are adequate"	0	1	2	13	4
"I became familiar with the recommender system quickly"	0	1	2	7	10
"I understood why the events were recommended to me"	0	1	5	9	5
"The recommender can be trusted"	0	0	11	9	0
"Overall, I am satisfied with the recommender"	0	0	7	11	2
"I would use this recommender again"	1	0	2	11	6
"I would buy tickets for events, given the opportunity"	0	1	2	8	9

	Positions of Top 3	Positions of Likes
Participant 1	10, 2, 3	1, 2, 10
Participant 2	5, 2, 3	2, 3, 5, 6, 7, 10
Participant 3	6, 8, 3	6, 8
Participant 4	1, 3, 9	1, 3, 9
Participant 5	3, 7, 2	3, 7
Participant 6	2, 3, 8	2, 3, 4, 7, 8
Participant 7	8, 9, 10	2, 4, 8, 9, 10
Participant 8	10, 9, 3	1, 2, 3, 4, 5, 6, 7, 9, 10
Participant 9	1, 5, 6	1, 2, 3, 4, 5, 6, 7, 9
Participant 10	2, 4, 10	1, 2, 4, 8, 10
Participant 11	9, 4, 2	2, 4, 5, 6, 7, 8, 9
Participant 12	6, 4, 10	1, 4, 6, 10
Participant 13	3, 4, 10	1, 2, 3, 4, 5, 7, 9, 10
Participant 14	5, 4, 6	4, 5, 6, 7
Participant 15	10, 2, 8	2, 3, 8, 10
Participant 16	3, 7, 1	3
Participant 17	7, 1, 9	1, 3, 5, 6, 7, 9, 10
Participant 18	2, 4, 1	1, 2, 4, 5
Participant 19	2, 3, 10	2, 3, 6, 10
Participant 20	6, 1, 8	1, 6, 8

Bibliography

[1] *4A Solutions*. URL: http://www.4a-solutions.de/ (visited on 07/01/2019).

[2] G. Adomavicius and A. Tuzhilin. "Context-Aware Recommender Systems." In: *Recommender Systems Handbook*. Springer, 2015, pp. 191–226.

[3] G. Adomavicius and A. Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." In: *IEEE Transactions on Knowledge & Data Engineering* 6 (2005), pp. 734–749.

[4] J. Akehurst, I. Koprinska, K. Yacef, L. Pizzato, J. Kay, and T. Rej. "CCR: A Content-Collaborative Reciprocal Recommender for Online Dating." In: *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[5] D. Alur, D. Malks, J. Crupi, G. Booch, and M. Fowler. *Core J2EE Patterns: Best Practices and Design Strategies*. Sun Microsystems, Inc., 2003.

[6] Apache. *Apache Mahout: For Creating Scalable Performant Machine Learning Applications*. URL: https://mahout.apache.org/ (visited on 07/01/2019).

[7] Apple. *Apple Developer Documentation: identifierForVendor*. URL: https://developer.apple.com/documentation/uikit/uidevice/1620059-identifierforvendor (visited on 07/01/2019).

[8] Apple. *Apple Developer Human Interface Guidelines: Pickers*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/controls/pickers/ (visited on 07/01/2019).

[9] Apple. *Apple Developer Human Interface Guidelines: Tab Bars*. URL: https://developer.apple.com/design/human-interface-guidelines/ios/bars/tab-bars/ (visited on 07/01/2019).

[10] Balsamiq Studios. *Balsamiq Wireframes: Quick and Easy Wireframing Tool*. URL: https://balsamiq.com/wireframes/ (visited on 07/01/2019).

[11] *Bandsintown*. URL: https://www.bandsintown.com/ (visited on 07/01/2019).

[12] D. Bansal. *Swift 4 Recipe: Self Sizing Table View*. URL: https://medium.com/@dushyant_db/swift-4-recipe-self-sizing-table-view-2635ac3df8ab (visited on 07/01/2019).

[13] V. R. Basili, G. Caldiera, and H. D. Rombach. "The Goal Question Metric Approach." In: *Encyclopedia of Software Engineering* (1994), pp. 528–532.

[14] D. M. Brown. *Communicating Design: Developing Web Site Documentation for Design and Planning*. New Riders, 2010.

[15] *Bumble*. URL: https://bumble.com/ (visited on 07/01/2019).

[16] R. Burke. "Hybrid Web Recommender Systems." In: *The adaptive web*. Springer, 2007, pp. 377–408.

[17] F. Buschmann, K. Henney, and D. C. Schmidt. *Pattern-Oriented Software Architecture, on Patterns and Pattern Languages*. Vol. 5. John wiley & sons, 2007.

[18] B. P. Buunk, P. Dijkstra, D. T. Kenrick, and A. Warntjes. "Age Preferences for Mates As Related to Gender, Own Age, and Involvement Level." In: *Evolution and Human Behavior* 22.4 (2001), pp. 241–250.

[19] D. Cherry. *SwiftHTTP: Thin Wrapper Around NSURLSession in Swift. Simplifies HTTP Requests.* URL: https://github.com/daltoniam/SwiftHTTP (visited on 07/01/2019).

[20] *CocoaPods*. URL: https://cocoapods.org/ (visited on 07/01/2019).

[21] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge, 2013.

[22] T. De Pessemier, J. Minnaert, K. Vanhecke, S. Dooms, and L. Martens. "Social Recommendations for Events." In: *CEUR workshop proceedings*. Vol. 1066. 2013.

[23] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, et al. "Collaborative Filtering Recommender Systems." In: *Foundations and Trends® in Human–Computer Interaction* 4.2 (2011), pp. 81–173.

[24] *Eventbrite*. URL: https://www.eventbrite.de/ (visited on 07/01/2019).

[25] *Eventguide*. URL: https://www.in-muenchen.de/vertrieb/in-muenchen-eventguide-app-konzerte-opern-ausstellungen-partys-open-airs-ios-android-8748566.html (visited on 07/01/2019).

[26] Federal Statistical Office of Germany. *In 41% aller Haushalte in Deutschland lebt nur eine Person*. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/Zahl-der-Woche/2017/PD17_31_p002.html (visited on 07/01/2019).

[27] J. J. Garrett. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. Pearson Education, 2010.

[28] Google. *Automatically Collected User Properties*. URL: https://support.google.com/firebase/answer/6317486 (visited on 07/01/2019).

[29] D. Herzog. "*Ein hybrider Algorithmus für mobile und kontextsensitive Veranstaltungsempfehlungen*." MA thesis. Technical University of Munich, July 2015.

[30] D. Herzog and W. Wörndl. "Extending Content-Boosted Collaborative Filtering for Context-Aware, Mobile Event Recommendations." In: *Proceedings of the 12th International Conference on Web Information Systems and Technologies*. 2016.

[31] B. P. Knijnenburg and M. C. Willemsen. "Evaluating Recommender Systems With User Experiments." In: *Recommender Systems Handbook*. Springer, 2015, pp. 309–352.

[32] I. Koprinska and K. Yacef. "People-to-People Reciprocal Recommenders." In: *Recommender Systems Handbook*. Springer, 2015, pp. 545–567.

[33] A. Krzywicki, W. Wobcke, Y. S. Kim, X. Cai, M. Bain, A. Mahidadia, and P. Compton. "Collaborative Filtering For People-to-People Recommendation in Online Dating: Data Analysis and User Trial." In: *International Journal of Human-Computer Studies* 76 (2015), pp. 50–66.

[34] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. "Event-Based Social Networks: Linking the Online and Offline Social Worlds." In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 1032–1040.

[35] *Lovoo*. URL: `https://www.lovoo.com/` (visited on 07/01/2019).

[36] A. Q. Macedo, L. B. Marinho, and R. L. Santos. "Context-Aware Event Recommendation in Event-Based Social Networks." In: *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM. 2015, pp. 123–130.

[37] H. B. Mann and D. R. Whitney. "On a Test of Whether One of Two Random Variables is Stochastically Larger Than the Other." In: *The annals of mathematical statistics* (1947), pp. 50–60.

[38] *MariaDB*. URL: `https://mariadb.org/` (visited on 07/01/2019).

[39] L. Masinter. *RFC7578: Returning Values from Forms: multipart/form-data*. URL: `https://tools.ietf.org/html/rfc7578` (visited on 07/01/2019).

[40] *Meetup*. URL: `https://www.meetup.com/` (visited on 07/01/2019).

[41] P. Melville, R. J. Mooney, and R. Nagarajan. "Content-Boosted Collaborative Filtering for Improved Recommendations." In: *Eighteenth national conference on Artificial intelligence*. AAAI. 2002, pp. 187–192.

[42] E. Minkov, B. Charrow, J. Ledlie, S. Teller, and T. Jaakkola. "Collaborative Future Event Recommendation." In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM. 2010, pp. 819–828.

[43] *Münchner Singles*. URL: https://www.muenchnersingles.de/ (visited on 07/01/2019).

[44] C. Nance, T. Losser, R. Iype, and G. Harmon. "NoSQL vs RDBMS - Why There is Room for Both." In: *Proceedings of the Southern Association for Information Systems Conference*. SAIS. 2013, pp. 111–116.

[45] R. E. Nicholson and D. G. Pearce. "Why Do People Attend Events: A Comparative Analysis of Visitor Motivations at Four South Island Events." In: *Journal of Travel Research* 39.4 (2001), pp. 449–460.

[46] T. Oliver. *TOCropViewController: A View Controller for iOS That Allows Users to Crop Portions of UIImage Objects*. URL: https://github.com/TimOliver/TOCropViewController (visited on 07/01/2019).

[47] A. F. Osborn. "Applied Imagination." In: (1953).

[48] L. Pizzato, T. Rej, J. Akehurst, I. Koprinska, K. Yacef, and J. Kay. "Recommending People to People: The Nature of Reciprocal Recommenders With a Case Study in Online Dating." In: *User Modeling and User-Adapted Interaction* 23.5 (2013), pp. 447–488.

[49] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay. "RECON: A Reciprocal Recommender for Online Dating." In: *Proceedings of the fourth ACM conference on Recommender systems*. ACM. 2010, pp. 207–214.

[50] L. Pizzato, T. Rej, K. Yacef, I. Koprinska, and J. Kay. "Finding Someone You Will Like and Who Won't Reject You." In: *International Conference on User Modeling, Adaptation, and Personalization*. Springer. 2011, pp. 269–280.

[51] L. Pizzato and C. Silvestrini. "Stochastic Matching and Collaborative Filtering to Recommend People to People." In: *Proceedings of the fifth ACM conference on Recommender systems*. ACM. 2011, pp. 341–344.

[52] P. Pu, L. Chen, and R. Hu. "A User-Centric Evaluation Framework for Recommender Systems." In: *Proceedings of the fifth ACM conference on Recommender systems*. ACM. 2011, pp. 157–164.

[53] Red Hat. *Hibernate ORM: Your Relational Data. Objectively.* URL: https://hibernate.org/orm/ (visited on 07/01/2019).

[54] P. Resnick and H. R. Varian. "Recommender Systems." In: *Communications of the ACM* 40.3 (1997), pp. 56–59.

[55] F. Ricci, L. Rokach, and B. Shapira. "Recommender Systems: Introduction and Challenges." In: *Recommender Systems Handbook*. Springer, 2015, pp. 1–36.

[56] S. Robertson and J. Robertson. *Mastering the Requirements Process: Getting Requirements Right*. Addison Wesley, 2012.

[57] L. C. Rong Hu and P. Pu. *ResQue*. URL: https://hci.epfl.ch/research-projects/resque/ (visited on 07/01/2019).

[58] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. "Collaborative Filtering Recommender Systems." In: *The adaptive web*. Springer, 2007, pp. 291–324.

[59] B. Smyth. "Case-Based Recommendation." In: *The Adaptive Web*. Springer, Berlin, Heidelberg, 2007, pp. 342–376.

[60] *SourceTree*. URL: https://www.sourcetreeapp.com/ (visited on 07/01/2019).

[61] Statistical Office of Munich. *Private Haushalte*. URL: https://www.muenchen.de/rathaus/Stadtinfos/Statistik/Bev-lkerung/Haushalte.html (visited on 07/01/2019).

[62] *Tinder*. URL: https://tinder.com/ (visited on 07/01/2019).

[63] J. M. Vigil. "Asymmetries in the Friendship Preferences and Social Styles of Men and Women." In: *Human Nature* 18.2 (2007), pp. 143–161.

[64] P. Xia, B. Liu, Y. Sun, and C. Chen. "Reciprocal Recommendation System for Online Dating." In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM. 2015, pp. 234–241.

[65] Xmartlabs. *Eureka: Elegant iOS Form Builder in Swift*. URL: https://eurekacommunity.github.io/ (visited on 07/01/2019).

[66] Xmartlabs. *Eureka: Repository*. URL: https://github.com/xmartlabs/Eureka (visited on 07/01/2019).