RMIT UNIVERSITY

Matchmaking System
Dating website D8Finder
Technical Solution Design

Version: V1.2
Date: 04/11/18
Sponsor: Reza Soltanpoor
Author: Morgan Sylven , Matthew Hezel, Moonyoung Jung, Jimin Ahn

Document Control

Distribution

Version	Issued	Recipient	Position
V 1.0	11/10/18	Reza Soltanpoor	Client
V 1.1	21/10/18	Reza Soltanpoor	Client
V 1.2	04/11/18	Reza Soltanpoor	Client

Amendment History

Section	Author	Page	Version	Comment
1	Morgan	3	1.0	Added introduction.
1	Morgan	4	1.0	Added technical environment.
4	Matthew	5	1.0	Addeed system architecture.
5	Matthew	6	1.0	Added database architecture.
6	Matthew	6	1.0	Added implementation instruction.
3	Matthew	4	1.0	Added overall architecture.
1	Morgan	4	1.0	Updated technical environment.
1	Morgan	3	1.0	Updated introduction.
8	Matthew	7	1.0	Added known issues & risks.
5	Jimin	6	1.0	Updated database architecture.
7	Moonyoung	7	1.1	Updated nonfunctional requirements.
9	Moonyoung	8	1.1	Added other considerations.
9	Moonyoung	8	1.2	Updated other considerations.

.

Staff or Entities Consulted

NAME	Position / Organization
Morgan Sylven	Developer/Scrum Leader
Matthew Hezel	Developer
Moonyoung Jung	Developer/UX Designer
Jimin Ahn	Developer/Product Owner

Table of Contents

1 Introduction

The D8 Finder website is a dating site made to match users for the purposes of pursuing romantic relationships. Users are matched based on shared interests, location, and age and are then given an opportunity to decide if they are interested based on further details in the user profiles; if both users decide they are interested they are then given a means to communicate through the site without needing to share private information.

The site is produced primarily using web languages. Cloud9 was used for collaborative coding.

The project is fairly simple in most regards with the messaging system and matchmaking algorithm being the most complex. Although not inherently difficult from a web development standpoint creating the website has proven to be rather complex due to the team's collective lack of experience with the languages.

The project exists to solve the problem of a lack of minimalist dating sites, the current pool of dating sites all require either a niche interest, a social media connection, or have too many features cluttering the experience. Our project exists to provide the online dating experience to users who want to meet people and form relationships without any distracting features or the hassles and privacy concerns inherent in social media use.

2 Technical Environment

Languages used: PHP, CSS, JQuery/JavaScript, HTML, MySQL.
Server hosting site: Ubuntu running Apache
IDE & Source Control: Cloud9
External components: Bootstrap, JQuery Validate

The site has been built using standard web languages. HTML and CSS were used for the front end as other options such as sass and less, were avoided to minimise complexity where it isn't needed, although it should be noted that Bootstrap uses SCSS, this is internal to Bootstrap and wasn't utilised elsewhere in the codebase. Javascript and JQuery are used for some client side operations although sparingly throughout the project in favour of PHP, even for client side operations. PHP is used as the only server side language, it was chosen over languages such as NodeJS and RoR due to its stability, robustness, and in depth documentation and support.

Cloud9 was used for the creation and control of the code, it is also the host of the web server, it was chosen due to its ability to host the live site as well as be a central repository of code. Cloud9's ability to allow multiple remote users to edit code simultaneously was a major contributing factor to its choice for all of these roles. Although not ideal for a major commercial project, smaller scale applications such as this make it a perfect choice.

The system utilises several external resources to complete and complement its' functionality. These include Bootstrap as the front-end framework; JQuery Validate, a jquery based data validation tool; PHPMailer, an email handling system with SMTP support;

3 Overall Architecture

The Architecture of the website consists of three layers. The first is the display layer which the user interacts with such as the login and dashboard as well as input into such as upload profile picture or changing password. The second lay is two connect the display layer and the database layer, process sql queries and store the profile image. The final layer the previously

mentioned database layer is incharge of storing the data of the uses and the name of the profile image.

4 System Architecture

4.1 FUNCTIONALITIES/FEATURES

4.1.1 Login

4.1.2 Logout

4.1.3 Register

4.1.4 Recover Forgotten Password

4.1.5 Change Password

4.1.6 Password Encryption

4.1.7 Admin Login

4.1.8 Admin Logout

4.1.9 User Dashboard

4.1.10 Admin Dashboard

4.1.11 Cancel or Delete Account

4.1.12 Verify Email

4.1.13 Edit User Dashboard

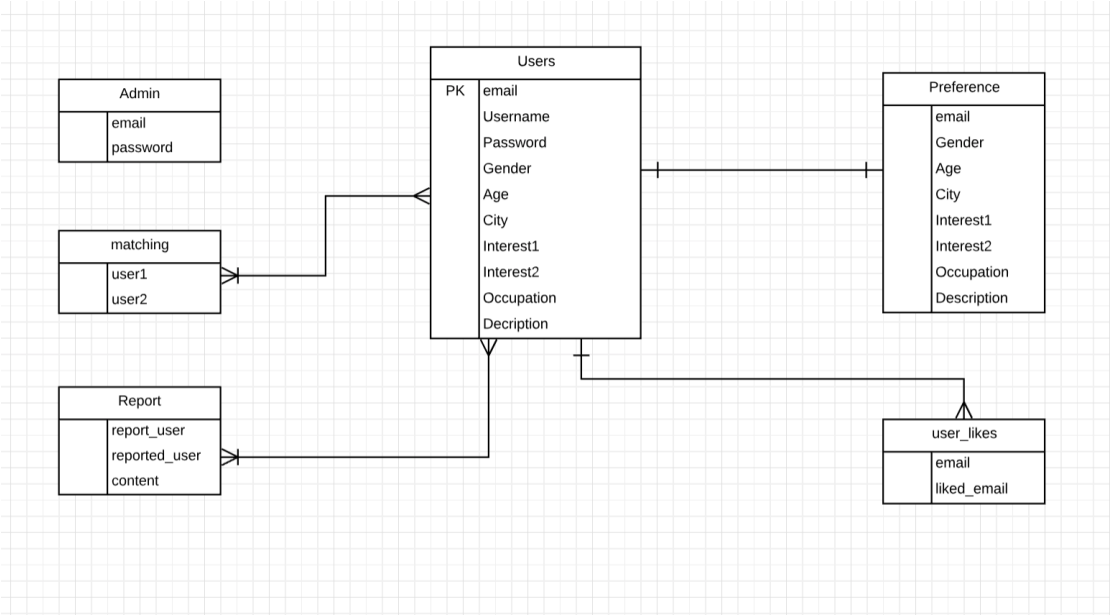
4.1.14 Edit User Profile

4.1.15 Upload Profile Image

4.1.16 Admin Delete Account

4.1.17 Messaging

5 Database Architecture



The database consists of five tables admin, matching, preference, users, user_likes. The admin table consists of three columns id, email, password. The matching table has three columns match_id, user1, user2. The nine columns for the preferences table are email, age_min, age_max, gender, city, education, occupation, interest1, and interest2. The fifteen columns for the users table are id, username, email, password, gender, age, city, education, occupation, general_info, astivationHash, activated, interest1, interest2, profile_image. The last table is user_likes with two columns email, liked_email.

This Database Architecture is designed to be interconnected and scalable without the admin adding in more rows and columns manually for theoretical an infinite number of people, but due to network limitation, physical storage and the current laws of physic means that this number will not be attainable.

6 Implementation Instructions

To make the website functional the owner will have to have to have the following server specs:

- cpu with at least four cores or more.
- 8GB of memory or more.
- network connectivity preferably one or more LAN ports.
- 1TB hard drive with the option for more.
- power supply that offer more watts than needed for expandability.

- motherboard that can support the above specs.

The website and its database are current holding temporary data so it is not needed to be migrated. If the system is replacing an old system while still use the old database (that is compatible) then database.php and server.php in the config folder as well as recommandation.2.php, recommandation.php, verify.php, and email_confirm.php will need to be changed in order to connect to the old database.

7 Non-functional requirements

QoS Attribute	Reason
ACCESS SECURITY	● User should be able to success the verification process via verification link which was sent by the system while registration. ● User password will not be viewable at the point of entry or at any other time.
AVAILABILITY	● The system will be able to accessible at anytime for any users unless there was exceptional circumstances.
CONFIDENTIALITY	● User password will be stored in encrypted version and not viewable at any point. ● User information will only be used when system use it for matching the users. ● Admin only able to delete users from the system if there is sufficient amount of report from other users.
USABILITY	● User match is occured when user choose Like and corresponding user also likes the user. ● The application only available to use by Adult (over 18).
MAINTAINABILITY	● A development programmer who supporting this software application will be able to add a new product feature, including source code modifications and testing, with no more than one month.

8 Known Issues & Risks

Known Issues & Risks include:

- Sql Injections in login page and other input areas.
- Forgot password does not decrypt the password just sends the hashed version.
- Only some of the user passwords are not hashed so password are checked using plain texted.
- No backup for database making data a real threat.
- Some function on the website do not load correctly and display warnings.

9 Other Considerations

In the design perspective, our application need to update layout and styling for better user experience and better looks. Code validation should be done for the better security and website performance.

10 Appendix

Technical User stories

1. Design and construct database to store website data
2. Make user guide documents and other documentation
3. Deploy website