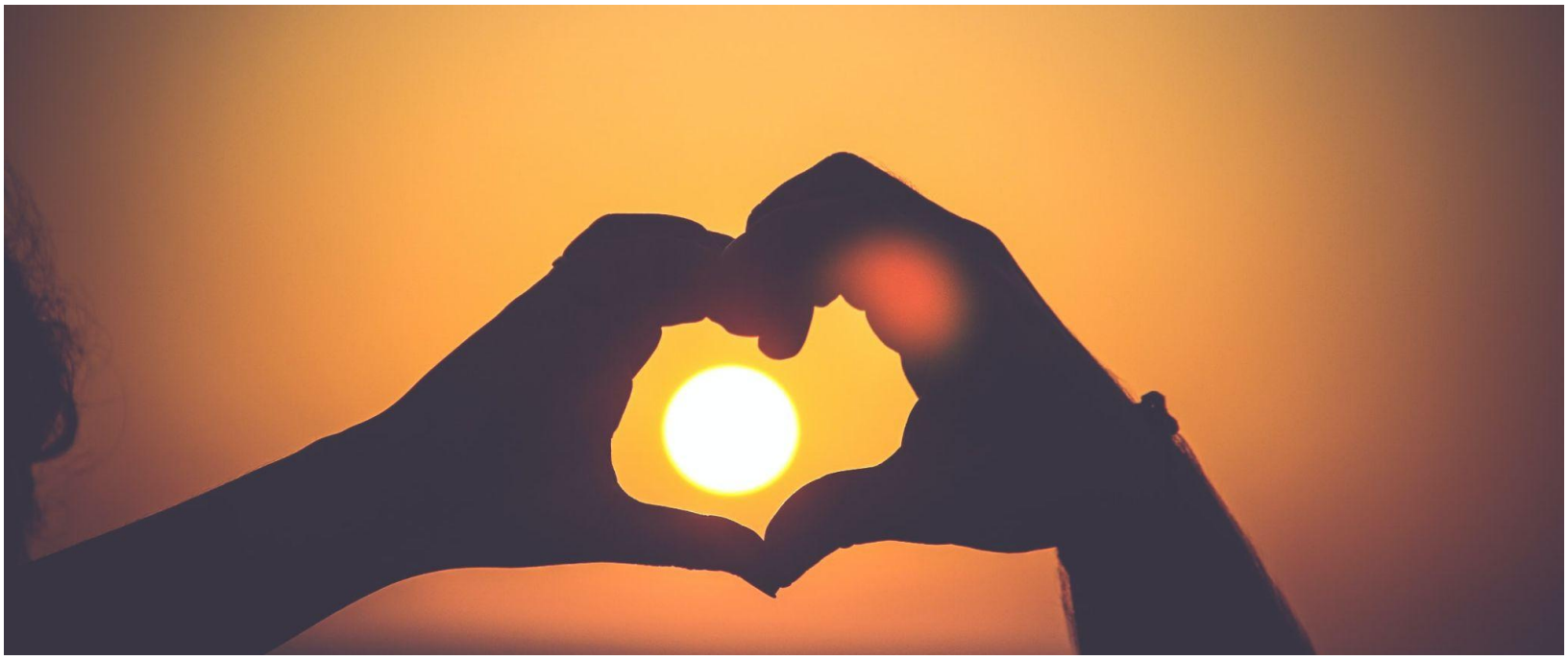
# CODE FIRST GIRLS

Summer 2022: Software Engineering Cohort

# DIY COUPLES THERAPY PROJECT REPORT



**Janet Osunsami**

**Kyla Lam**

**Lucy Hawkins**

**Reema Ramrakka**

# INTRODUCTION

## Aims and objectives of the project

Our site aims to facilitate DIY(Do-It-Yourself) couples therapy. It allows two users (ideally a couple) to generate and input suggestions that their partner could implement to improve their relationship. These suggestions will generate actionable results i.e. links to external reading, youtube videos, ibooks, Amazon wishlists, groupon etc. Users can specify that suggestions are returned from one or more of the 5 love languages.

The 5 love languages are:

- Words of affirmation
- Acts of service
- Gift giving
- Quality time
- Physical touch

For example: if you want your partner to improve on quality time, you select that language and the resulting suggestion could be a link to a skydiving website. We added suggestions and used a randomizer to gamify the activity for users.

NHS mental health services and counselling lists currently boast over year-long wait times and private therapy can set couples back anywhere from £60-£250+ per session. This, in combination with the social awkwardness resulting from multiple lockdowns and the cost of living crisis, may make confronting relationship issues without support, rather daunting.

Standards are rising, particularly in heterosexual relationships, and dating opportunities for heterosexual men are diminishing as a result. A recent study states that men need to address skills deficits to meet healthier relationship expectations.

Our DIY couples therapy site allows couples to generate problem-solving actionable results that could save relationships. Sometimes telling your other-half exactly what you need or what would help is difficult to confront and articulate. Rather than writing out a demanding, extensive list of dissatisfaction, why not gamify the whole situation? Our site removes the direct confrontational aspects of communication and offers a subtler yet equally effective solution. It provides an alternative way to work through relationship problems and gain a better understanding of each other in a fun way.

**Roadmap**


DIY COUPLES THERAPY
CFG PROJECT
JANET O, KYLA L, REEMA R, LUCY H
TASK
18TH JUL
25TH JUL
1ST AUG
8TH AUG
Project Theme Selection
Jul 18 - 20
Define Goals & Purpose
Jul 21 - 22
Backend Design - Key Features
Jul 25 - 27
Love Language Suggestions
Jul 28 - 30
Key Features Implementation
1st Aug - 10th
Mentors Feedback
Aug 12
Unit Tests
Aug 10 - 11
Web Design
1st Aug - 10th
Review
Aug 13
Report
Aug 10 - 13

# BACKGROUND

Our website provides a love language suggestion tool that not only produces randomised suggestions as an output, but also allows users to input activities for each love language as well. The website will include a randomizer, a suggestion input feature, a suggestion output feature, a love language quiz, and a word cloud API that showcases the most used love language activities for each love language.

As the user enters the website, they will select the type of love language. If they do not know what their love language is, they will be directed to the Love Language Quiz that is also displayed on the front page. The user will then click through different love language suggestions suggested by not only the DIY Couples Therapy team, but also other users. If a user wishes to input a new suggestion to the website, they will have to specify which love language their suggestion corresponds to, as well as their name so we can display it. Adding a link to compliment the suggestion is optional. This data (suggestions, names, links) will be stored in the SQL database. This will update the word cloud art on our website for each love language to display the most popular love language activities. As the database grows, the word cloud art will adjust accordingly.

# SPECIFICATIONS AND DESIGN

### Requirements technical and non-technical

For the Love language selector, users are able to specify that they would like actionable results to be generated for their partner within a specific category. The results will be randomly generated in the back-end within each specific category when users specify a love language. For example, the back-end for the love language ‘Acts of Service’ could consist of the following options:

- A breakfast recipe and shopping list, encouraging them to prepare breakfast for their partner before work
- A link to purchase a house-cleaning voucher
- A link to a car-wash voucher
- Order their favourite deliveroo order

If a user's partner specifies 'Acts of service' as their love language, the back-end will select an option from the above database at random.

For the Suggestion inputter, both developers and users can add new suggestions, with or without links and specify which category of love language they fit into. Users can also state their name so we can assign credit to a great new suggestion.

For the home page there is a link displayed to the Love language quiz so users can discover their love language, if they don’t already know it.

We need all of these functions to have quick recall time. The site must transition smoothly throughout the user journey.

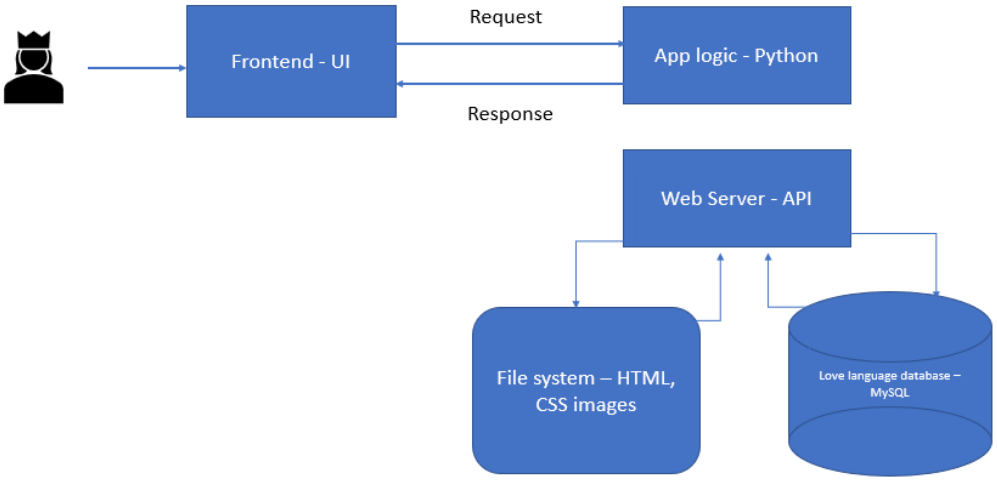
The front end needs to be visually appealing.

The suggestions need to be random and they need to vary greatly from love language to love language. There should be minimal, if any, crossover.

The love language word clouds generated by both the users' and the site's suggestions should be informative, inspiring, accurate and visually appealing. They also need to update as the suggestions do.

### Design and architecture

Originally, we weren't entirely sure how the architecture of our site would be compiled but thought it would look something like the below figure:




**We used the following languages and tools to build our site:**

**Word cloud API** - to display word clouds that inspire users to take and make great love-language based suggestions.
**Flask** - to build this app's web application.
**Python** - connecting MySQL to Python + key features of the system.
**MySQL-connector** - to connect the database to the python code.
**Requests** - for the HTTP library.
**MySQL** - database for actionable items/suggestions and users' details.
**HTML** - to build the front end.
**CSS** - to make the front end visually appealing.
**Bootstrap** - for simplified, expedited CSS implementation.

# IMPLEMENTATION AND EXECUTION

### Development approach and team member roles

We began by creating a trello board to map out specifications and roles:
https://trello.com/invite/b/RXj1G3hp/77aa58e0bd3f3aa6b2a3815ec3aa6dad/suggestion-solution-site

We also created a quick spreadsheet to divide up tasks from the get-go. It looked like this:

**Overall tests** - Janet
**Key features of app: randomiser, love language selector** - Kyla
**Key features of app: suggestion inputter, suggestion outputter** - Reema
**Key features of app: Home/login page, love language quiz** - Lucy
**API input** - Lucy
Love language suggestions with links - All

Lucy started on the base code for our site, joining an app.py file to a helpers.py file to several html files and a CSS stylesheet. We used flask, requests initially and then also a SQL connector.

We all contributed to the suggestions for the love language suggestions list and tried to be as creative as possible! Here's the first iteration of our suggestions list. We have subsequently added 'user generated' suggestions at the bottom of this sheet and imputed them to our site. As our site is not hosted nor live, we had to create the user generated content ourselves at this stage. However the input section is fully functional and users can input new suggestions that are then stored in our SQL database.

Janet worked on an HTML-based comment inputter where users could write their love language down and then return to it. But as we abandoned the login and user profile ideas, this too was abandoned.

Kyla and Reema worked in parallel on the randomiser and the suggestion outputter and did a mass of data entry also to get all of our suggestions in pycharm. As we pivoted and moved the data over to SQL, the task had to be repeated.

This final task list delegation then looked like the below:

**SQL connector/ input feature** - Reema/ Kyla
**Unit tests** - Janet/ Kyla
**Front-end** - Lucy/All
**Classes in output feature** - Reema
**Report** - All
**PowerPoint** - All

## Tools and libraries

We used PyCharm as our IDE in Python, and libraries such as mysql.connector, requests, Flask, render_template, redirect; we also used mySQL for SQL, and HTML, CSS and Bootstrap for web development.

## Implementation process

## Challenges/Changes

1. Initially, we were going to store the database in two different environments - mySQL and PyCharm. Then we decided to solely use mySQL to store data and connect it to PyCharm.
2. Initially, we were going to create a login system for individual users to encourage engagement. Then we decided to overthrow that and instead create an input feature in an open submission format, so that everyone can input their suggestions to the database while keeping their names linked to the activity.

### Achievements

1. We added word cloud art that is linked to each love language in the database into the website.
2. Users are able to input love language suggestions into the database with their names displayed when the new suggestions are generated for other users. They are also able to include optional links.
3. Users are able select different love language suggestions accordingly, with or without additional links to support their suggestions - saving future users from googling a solution.

### Agile development

We took an iterative approach for our project where, throughout our four weeks of development, there were many instances of modifying and enhancing different aspects of deliverables - especially in the 'Key features implementation' and 'Mentors feedback' phases. Examples of modification included: how/where to store our suggestions databases.

Refactoring - restructuring of our classes python code, building upon the suggestion input feature by creating a 'submit a report' feature.

Code reviews - It was identified that the suggestion outputter would require a cache in order to successfully create a love language base class with five child classes (each implementing a unique love language); this was inefficient and bad code as it had over twenty-five lines of code for one functionality. After review, it was decided that we should use SQL to store the suggestion database. Love language tables were created in SQL which contained the love language name, id, and description.

## TESTING AND EVALUATION

Our testing strategy is using manual testing at API and UI levels.

**Home.html:**

- The URL connects to the Love Language Quiz
- Each love language selector button links to their separate love language suggestions
- The webpage is visually pleasing

**System limitation:**

- More images could be added for better presentation and user experience
- A short description of the purpose of the website could be added, rather than jumping right into choosing love language suggestions

**Love_language.html:**

- Love language suggestions are displayed nicely, including the names of suggesters and the default DIY Couples Therapy in a carousel format
- It successfully connects users to suggestion page if they desire to add suggestions to their chosen love language
- It successfully connects users to home page if they desire to explore different love language of their choice

**System limitation:**

- It would be nice to include multimedia format such as video or photos to enhance visualisation

- There is no prevention of overlapping or quality control if every user is allowed to input all that they want

**Suggest.html:**

- The web page allows users to input their name, and suggestion to their chosen love language
- It displays functional “Cancel” and “Submit” button, where users can either cancel their suggestion submission or proceed with their suggestion submission
- When testing in the corresponding love language suggestions page, it displays the exact inputs from the previous submission

**System limitation:**

- There is no button to delete or modify submission, which could easily disrupt the neatness of the suggestion list
- There is no warning that the submission will be displayed in public after submission, so users may not be prepared for that

**Data.html:**

- It displays word cloud art for each of the five love languages, aggregated from the love language database
- The word “partner” is the most used word in the suggestion database, followed by other words displayed in various colours
- The word cloud art updates as new suggestions are being inputted

**System limitation:**

- It may not show the most relevant words, rather it shows more stop words (eg. ‘the’, ‘a’)
- Users have to scroll down to view the word cloud art of each love language, which is not very intuitive. It could be improved by having the art side-by-side.
- The web pages are not yet tested for colour-blind users, which could limit accessibility.

# CONCLUSION

In this project, we have learnt to use Python and SQL to create a functioning website that stores new and existing data in a database. We also learnt to apply front-end web development using HTML in order to create an attractive platform. We have conducted various stages of specification and design, implementation and execution, and testing and evaluation. The project outcome shows 5 love languages where users are able to receive and insert suggestions according to their unique love language, to ultimately improve the relationship with their partner. Although there were setbacks, such as changes to the design and choosing a different approach, it was a quick and uniform decision amongst the team to make appropriate adjustments accordingly. Despite being under time constraint, our team understood one another’s strengths and weaknesses and worked incredibly well together to produce the final result.