Pear-to-Peer

Project Software Engineering
Group I
28th Jun 2024

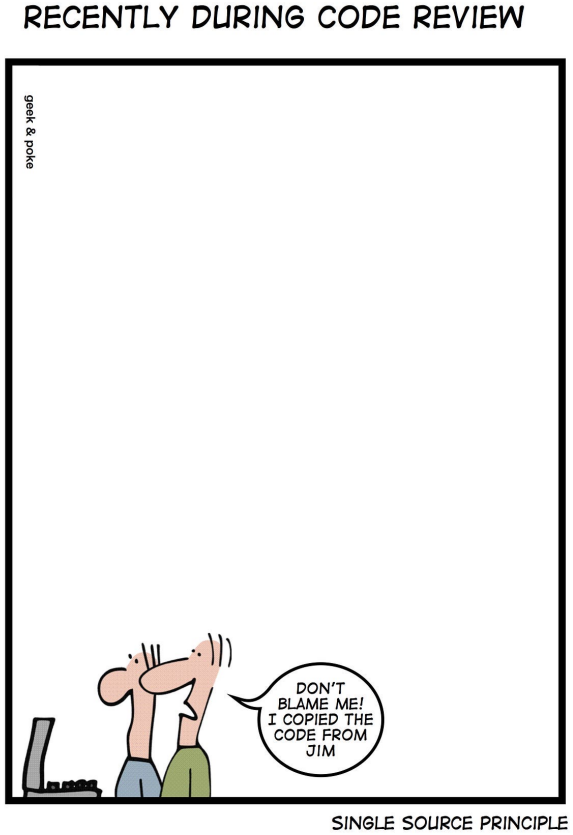
Why peer review and code review?

Importance of peer review in academia

- Filter for high-quality research
- Enhances credibility and prevents errors and fraud
- Professional development for the author

Importance of code review in software engineering

- Ensure code quality
- Enhances coding skill through constructive feedback



Pear-to-Peer

Platform that provides opportunities for students to enhance their academic feedback and coding skill, both as a reviewer and a reviewee.

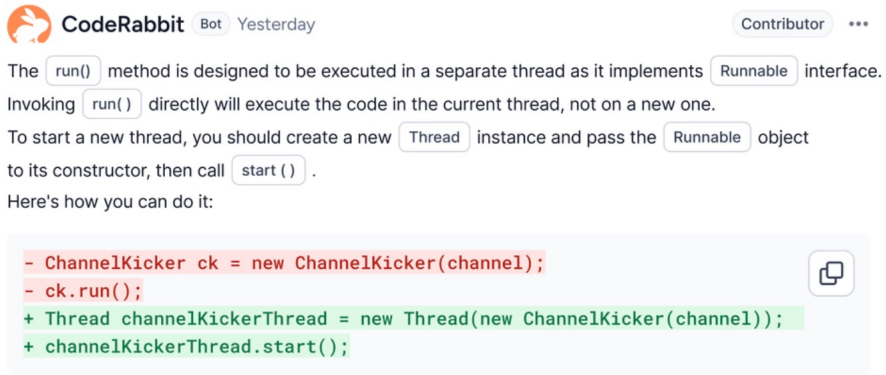
Uses LLM to provide high quality hint that help students to build constructive feedback

Why Pear-to-Peer?

Focus not only focus on the quality of the code, but also on the progress of educating student to code review and peer review

General LLM response does not match the level of the student

Prepares student for professional environments in the future job market



Team & Roles

Mentor: Jennifer van Langen & Frederick Kreuk

Group	Name	Goals
Backend	Max, Martijn, Janou, Adam	Database, backend API’s, user credentials, and security
Frontend	Kevin, Duco, Timon, Marijn	Website interface, design in Figma, routing
Server	Marijn	Deploy website on server
LLM	Marijn, Timon	Input and responses from OpenAi LLM
Git Master	Janou	Manage Git repo, fix conflicts
Scrum Master	Adam	Organize daily standup, and sprint of each week

Backend:

Server:

Frontend:

LLM:

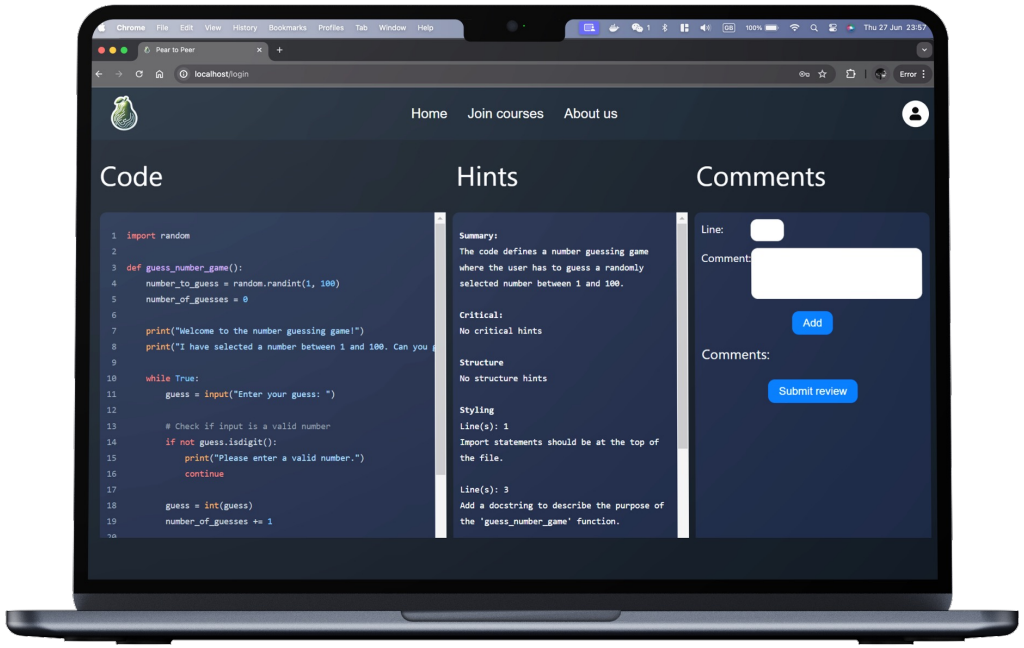
Version Control:

Minimum Viable Product

A website where the user can be either the submitter or reviewer

The submitter uploads code, and the software uses LLM models to summarize the functionality of the code and return hints to the parts that might have a problem

The reviewer writes feedback based on the output of LLM and sends it back to the submitter.



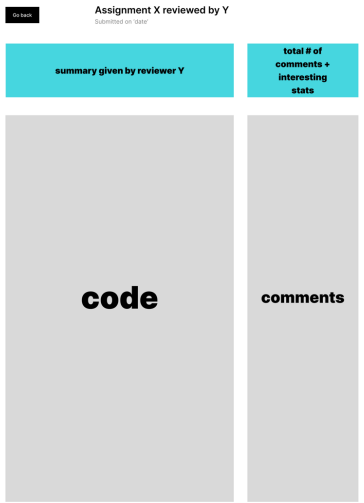
Frontend Design

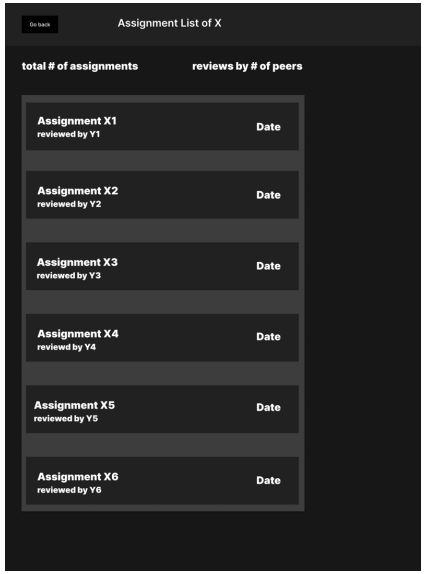
Dark mode

- Green coding: Energy efficient
- Reduce eye strain and protect sight
- Match to CodeGrade

Easy to use

- Minimize complexity by reducing buttons





App	30%	50%	100%
Calculator	21%/7%	36%/12%	81%/54%
Google Phone	18%/4%	34%/9%	81%/50%
Google Calendar	21%/6%	34%/16%	79%/57%
Google Maps	8%/1%	15%/4%	40%/19%
Google News	11%/1%	22%/5%	64%/29%
YouTube	12%/1%	21%/6%	56%/28%
Average	**15%/3%**	**27%/9%**	**67%/40%**

Table: Power reduction with dark mode at different brightness level for Google Pixel 3 (display reduction / total reduction) (1)

#888888 #1D9BF0 #192734 #15202B

(1): Pranab Dash and Y. Charlie Hu. 2021. How much battery does dark mode save? an accurate OLED display power profiler for modern smartphones.

LLM Design

GPT3.5 as more affordable choice

Specific query to locate and categorize the errors

```
You are an assistant designed to help users peer-review code by providing a summary and hints.
Your goal is to facilitate learning and improvement by pointing out potential issues related to
three different classes: critical errors, structure errors, and styling errors.

Critical errors include syntax errors, runtime errors, and logical errors.
Structure errors include poor modularization, inefficient data structures or algorithms, and improper use of co
Styling errors include poor naming conventions, lack of comments and documentation, and inconsistent indentatio

Provide hints using line numbers in this format:
>>> [line number(s)]: [suggestion]

Example:
Summary:
[summary of the code]

Hints with line numbers:
(Critical)
>>> 15: [suggestion]

(Structure)
>>> 1: [suggestion]

(Styling)
>>> 3-12: [suggestion]
```

LLM Design

Extra row number as input since GPT3.5 cannot recognize it

```
# 26 /* Adds inputted value to database if there is no such value in it.
# 27    It also prints out if the addition is done or why it fails.
# 28    Function either returns an '1' if it was not added or '0' if it was added.
# 29 */
# 30 int add(char *str) {
# 31     for (int index = 0; index < DB_SIZE; index++){
# 32         if (strcmp(str, DATABASE[index]) == 0){
# 33             printf("\"%s\" not added; already in database\n", str);
# 34             return 1;
# 35         }
# 36     }
# 37
# 38     if (DB_SIZE == MAX_DB_SIZE) {
# 39         printf("\"%s\" not added; database is full\n", str);
# 40         return 1;
# 41     }
# 42
# 43     char* var = malloc((strlen(str)) * sizeof(char*));
# 44     strcpy(var, str);
# 45     DATABASE[DB_SIZE++] = var;
# 46
# 47     printf("\"%s\" added to database\n", str);
# 48     return 0;
# 49 }
```

LLM Design

Return summary of code and hints as a .txt file

```
Summary:
The code implements a program that uses the Levenshtein algorithm to analyze inputted values and s
It provides functions to add, read, list, delete, compare, retrieve, get size, and handle commands
The main function processes user commands until quitting.

Hints with line numbers:
(Critical)
>>> 43: The memory allocation for the variable 'var' is incorrect.
    It should be malloc((strlen(str) + 1) * sizeof(char)) to include space for the null terminator
>>> 97: There is an out-of-bounds access when shifting values in the delete function. Change the c
>>> 102: There is a logical error when shifting values in the delete function. The loop condition

(Structure)
>>> 107: The function clean_grid is meant to free memory for a 2D array, but the parameter should
>>> 148: In the Levenshtein calculation, the initialization should be outside the loop since it ne
>>> 156-161: Consider breaking down this calculation using helper functions for better readability
>>> 198-211: The retrieval function mixes tasks (finding perfect match and calculating distances).
    Consider breaking it into smaller, more focused functions.

(Styling)
>>> 31: Add comments to explain the purpose of the loop in the add function.
>>> 64: Add comments to explain the purpose of each block in the read function.
>>> 127: Add more detailed comments explaining the steps of the Levenshtein algorithm.
>>> 171: Add comments to explain the purpose of the loops in the compare function.
>>> 188: Add comments to describe the logic behind the retrieval function.
>>> 265: Add more descriptive comments to explain the purpose of the functions and improve readabi
>>> 280: Improve the naming of arguments in the cmd function for clarity.

```

Green Coding & Testing

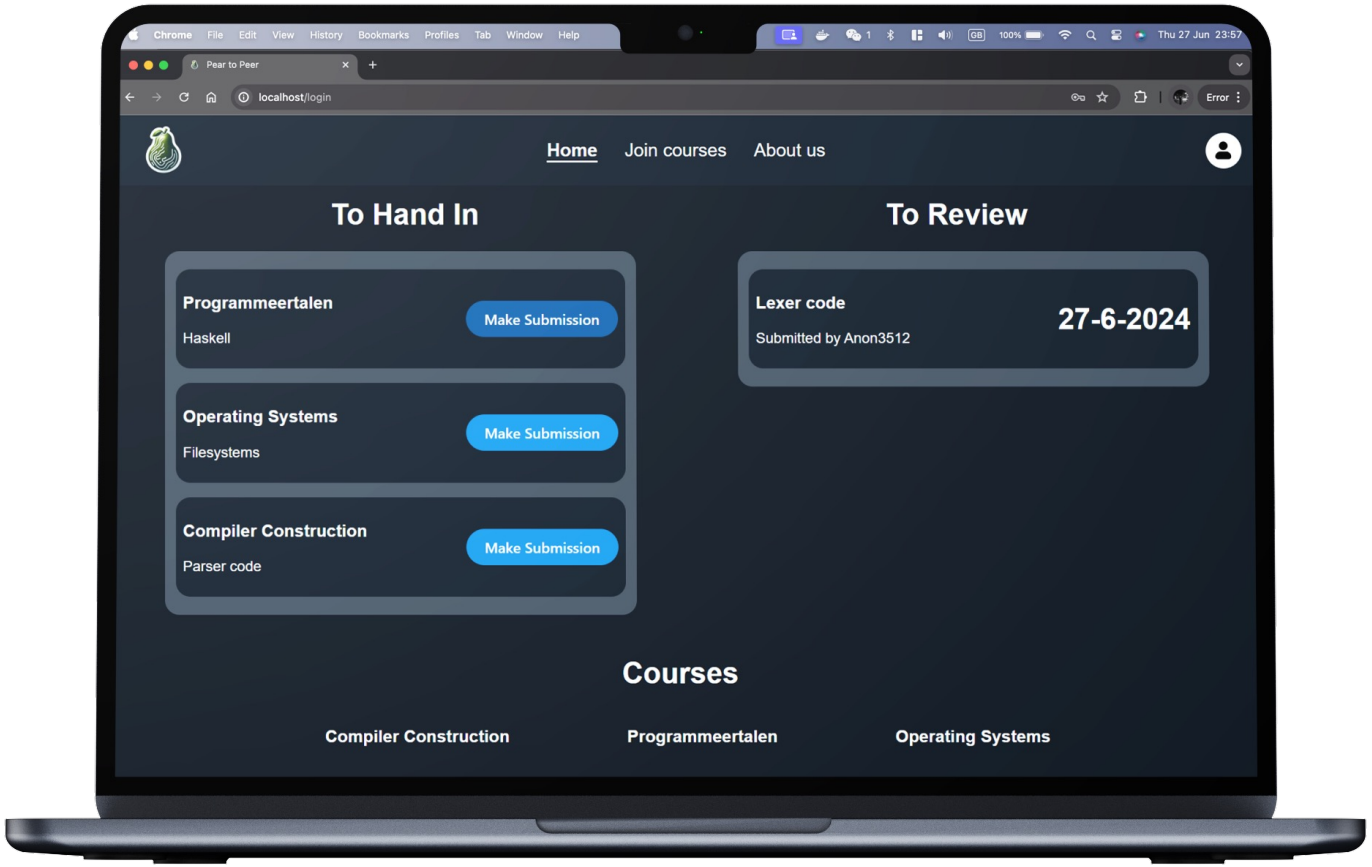
Frontend colour choice

Backend lightweight Flask and SQLite

Possible choice for LLM from OpenAI: remember conversation

Unit test but hard to write test script, due to complexity in the modules

Demo


Pear to Peer
localhost/login
Home
Join courses
About us
To Hand In
Programmeertalen
Haskell
Make Submission
Operating Systems
Filesystems
Make Submission
Compiler Construction
Parser code
Make Submission
To Review
Lexer code
Submitted by Anon3512
27-6-2024
Courses
Compiler Construction
Programmeertalen
Operating Systems

Future Potential

Personalize to different assignments specific to the academic education and assignments

Provide different level of responses of the LLM based of the choice of teacher

Skill tracking that allow student to track their progress and improvement

Analytics on student performance and common coding mistakes

Thank you

Project Software Engineering
Group I
28th Jun 2024