

CodeView



June 7, 2024

Student:

Adam, Janou, Max, Martijn, Finn, Marijn,
Kevin, Duco, Timon

Tutor:

Frederick
Jennifer

Group:

Group I



Inhoudsopgave

Frontend

Backend

LLM/AI



Planning



Progress

- React: Components!
- Create-react-app: No
- Remix: Yes

Figma draft

Assignment List of X

Go back

total # of assignments	reviews by # of peers
Assignment X1 reviewed by Y1	Date
Assignment X2 reviewed by Y2	Date
Assignment X3 reviewed by Y3	Date
Assignment X4 reviewed by Y4	Date
Assignment X5 reviewed by Y5	Date
Assignment X6 reviewed by Y6	Date

Assignment X reviewed by Y

Submitted on 'date'

Go back

summary given by reviewer Y

total # of comments + interesting stats

code

comments

Moving forward

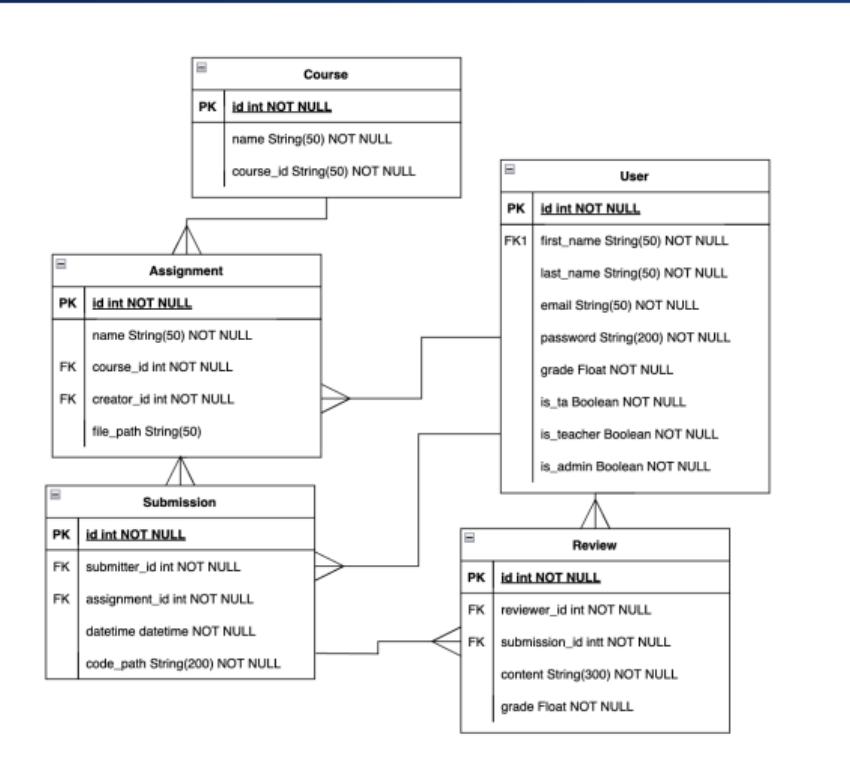
- Catch up with backend
- Connect to backend
- Add code linting
- More CSS

Database

Different tables in our database

- User
- Assignment
- Review
- Submission
- Course

Database



Testing

Users

First Name	Last Name	Email	Password	Is TA	True	False	Delete
Adam	Dong	adam2256@icloud.com		False	True	False	Delete
Form Fields:							
First Name:	<input type="text"/>						
Last Name:	<input type="text"/>						
Email:	<input type="text"/>						
Password:	<input type="password"/>						
Is TA:	<input type="checkbox"/>						
Is Teacher:	<input type="checkbox"/>						
Is Admin:	<input type="checkbox"/>						
<input type="button" value="Add User"/>							

Input

Part of Levenshtein code from Timon's IP assignment.

```
27 /* Adds inputted value to database if there is no such value in it.  
28 It also prints out if the addition is done or why it fails.  
29 Function either returns an '1' if it was not added or '0' if it was added.  
30 */  
31 int add(char *str) {  
32     for (int index = 0; index < DB_SIZE; index++){  
33         if (strcmp(str, DATABASE[index]) == 0){  
34             printf("\'%s\' not added; already in database\n", str);  
35             return 1;  
36         }  
37     }  
38  
39     if (DB_SIZE == MAX_DB_SIZE) {  
40         printf("\'%s\' not added; database is full\n", str);  
41         return 1;  
42     }  
43  
44     char* var = malloc((strlen(str)) * sizeof(char*));  
45     strcpy(var, str);  
46     DATABASE[DB_SIZE++] = var;  
47  
48     printf("\'%s\' added to database\n", str);  
49     return 0;  
50 }
```



Query

We turn GPT3.5 into an assistant with a specific query to locate errors in the code and categorize them by severity accordingly:

```
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 control structures.  
Styling errors include poor naming conventions, lack of comments and documentation, and inconsistent indentation.  
  
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]
```

Numbered Input Code

Since GPT3.5 cannot find the line number corresponding to the feedback by counting itself, we need to number the input code explicitly.

```
27 # 26 /* Adds inputted value to database if there is no such value in it.
28 # 27   It also prints out if the addition is done or why it fails.
29 # 28   Function either returns an '1' if it was not added or '0' if it was added.
30 # 29 */
31 # 30 int add(char *str) {
32 # 31   for (int index = 0; index < DB_SIZE; index++){
33 # 32     if (strcmp(str, DATABASE[index]) == 0){
34 # 33       printf("|%s| not added; already in database|\n", str);
35 # 34       return 1;
36 # 35     }
37 # 36   }
38 # 37   if (DB_SIZE == MAX_DB_SIZE) {
39 # 38     printf("|%s| not added; database is full|\n", str);
40 # 39     return 1;
41 # 40   }
42 # 41
43 # 42   char* var = malloc((strlen(str)) * sizeof(char*));
44 # 43   strcpy(var, str);
45 # 44   DATABASE[DB_SIZE++] = var;
46 # 45
47 # 46   printf("|%s| added to database|\n", str);
48 # 47   return 0;
49 # 48
50 # 49 }
```



GPT3.5 Output

This gives us a .txt file as output with the summary of the code and hints for each of the three categories.

```
1 Summary:  
2 The code implements a program that uses the Levenshtein algorithm to analyze inputted values and stores them in a database.  
3 It provides functions to add, read, list, delete, compare, retrieve, get size, and handle commands.  
4 The main function processes user commands until quitting.  
5  
6 Hints with line numbers:  
7 (Critical)  
8 >>> 43: The memory allocation for the variable 'var' is incorrect.  
9     It should be malloc(strlen(str) + 1) * sizeof(char)) to include space for the null terminator.  
10 >>> 97: There is an out-of-bounds access when shifting values in the delete function. Change the condition to 'idx < DB_SIZE - 1'.  
11 >>> 102: There is a logical error when shifting values in the delete function. The loop condition should stop at 'DB_SIZE - 1' instead of 'DB_SIZE'.  
12  
13 (Structure)  
14 >>> 107: The function clean_grid is meant to free memory for a 2D array, but the parameter should be int** instead of int**.  
15 >>> 148: In the Levenshtein calculation, the initialization should be outside the loop since it needs to be done only once.  
16 >>> 156-161: Consider breaking down this calculation using helper functions for better readability and maintainability.  
17 >>> 198-211: The retrieval function mixes tasks (finding perfect match and calculating distances).  
18     Consider breaking it into smaller, more focused functions.  
19  
20 (Styling)  
21 >>> 31: Add comments to explain the purpose of the loop in the add function.  
22 >>> 64: Add comments to explain the purpose of each block in the read function.  
23 >>> 127: Add more detailed comments explaining the steps of the Levenshtein algorithm.  
24 >>> 171: Add comments to explain the purpose of the loops in the compare function.  
25 >>> 188: Add comments to describe the logic behind the retrieval function.  
26 >>> 205: Add more descriptive comments to explain the purpose of the functions and improve readability.  
27 >>> 280: Improve the naming of arguments in the cmd function for clarity.  
28  
29
```

The main problem is that GPT3.5 is still inconsistent and will give different output upon another submission of the same code.