

## **Team Three: Final Report**

Michele Cook, Chelsea Nieves, Valerie Rudich, Constance Sturm, and Hoi Lam Wong

University of Maryland Global Campus






CMSC 495-7382: Capstone in Computer Science

Professor David Castillo

December 12, 2023

## Team Three: Final Report

### Table of Contents

1.	Overview	2
1.1.	Introduction	
1.2.	Individual Contributions	
2.	System Specification	2
3.	Project Plan	2
3.1.	 CMSC 495 Team 3 Project Plan	
4.	User's Guide	2
4.1.	 CMSC 495 - User Guide	
5.	Test Plan and Results	2
5.1.	 CMSC 495 - Test Plan	
5.2.	 CMSC 495 - Actual Test Cases	
6.	Requirements Specification (Project Design)	3
6.1.	 CMSC 495 Team 3 Project Design	
7.	Design and Alternate designs	3
8.	Development History	3
9.	Conclusion	8
9.1.	Lessons Learned	
9.2.	Design Strengths	
9.3.	Limitations	
9.4.	Suggestions	
10.	GitHub Link	10
10.1.	<a href="https://github.com/squanchthis13/Fortune-Teller">https://github.com/squanchthis13/Fortune-Teller</a>	

## **1.0 Overview**

### **1.1 Introduction**

This fortune teller application will reveal a fortune to you after you select if you want to login, register, or play as a guest. If you login you will be able to view past fortunes that you've saved. We hope you will enjoy using the application !

### **1.2 Individual Contributions**

Michele Cook	:	Documentation, and Testing
Chelsea Nieves	:	Design, Documentation, Development, Testing,
Valerie Rudich	:	Documentation, Development
Constance Sturm	:	Design, Documentation, and Development
Hoi Lam Wong	:	Design, Documentation, Development

## **2.0 System Specification**

Operating System: Window 10 +, Mac Sonoma 14

Programming Language: Python3

Additional Libraries: bcrypt

## **3.0 Project Plan**

Link: [☰ CMSC 495 Team 3 Project Plan](#)

## **4.0 User's Guide**

Link: [☰ CMSC 495 - User Guide](#)

## **5.0 Test Plan and Results**

Test Plan Link: [☰ CMSC 495 - Test Plan](#)

Test Results Link: [☰ CMSC 495 - Actual Test Cases](#)

## **6.0 Project Design/ Requirements Specification**

Link: [📄 CMSC 495 Team 3 Project Design](#)

## **7.0 Design and Alternate designs**

Our initial design was to design a program that is running in the console during Phase 1. We had to create a new design for GUI and while we did not have the time for a frontend mock-up first, our design at Phase 2, the users were able to sign in, register and receive a fortune message through buttons. Our design and GUI received a huge upgrade during phase 3, where we improved the flow and users now have easier access to play, sign in and register. We also included the ability to show different options to users who are guests (not logged in) and users who are logged on the main menu. The logged in user has the additional ability to view past fortunes they have received and saved. Previews of the different designs are included below in 8.0 Development History.

## **8.0 Development History**

---

November 17, 2023

Created Github repository. (Constance)

---

November 17, 2023 - November 21, 2023

Phase 1

- Added in skeleton code for the project (Constance & Chelsea)
    - Included the functionality to prompt users to either play, view rules, or exit.
  - Created .txt documents to hold fortunes to be displayed to the user. (Constance, Chelsea, Hoi)
-

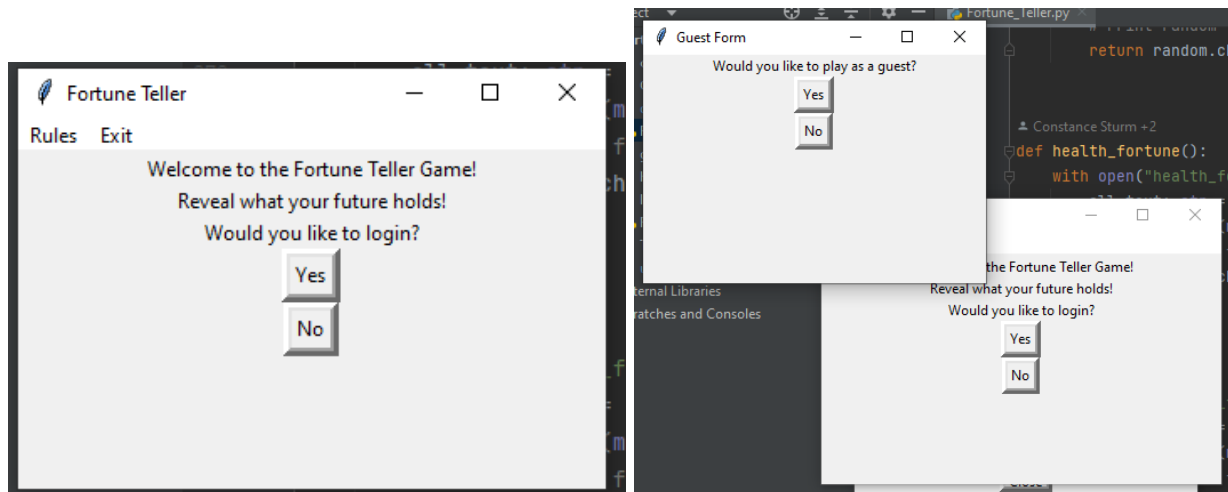
November 24, 2023 - November 28, 2023

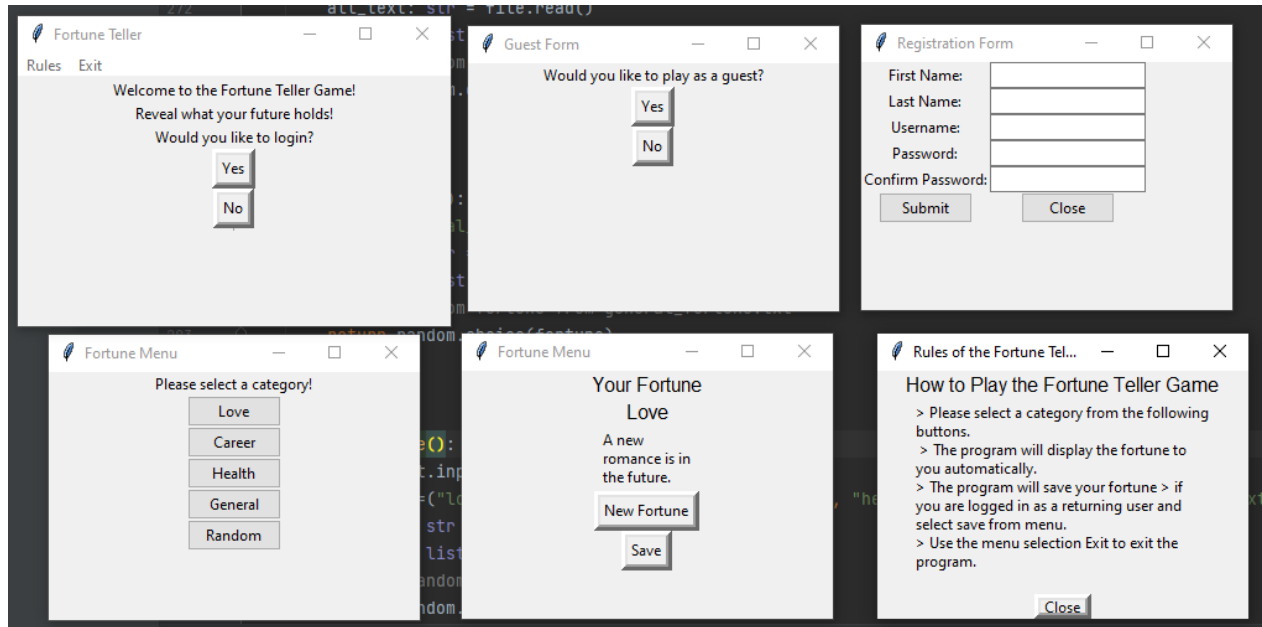
## Phase 2

Change to include GUI and to use SQL for saving fortunes system ( feedback from professor ) 11/25/2023

- 11/26/2023: Added in skeleton code for project with GUI (Constance)
  - Includes:
    - Main Menu, Fortune Menu
    - Registration Form, Rules
    - Menu bar on Main Menu
- 11/27/2023: Modified registration form, fixed bugs, refactoring codes (Hoi Lam)
  - Included a main method to run whole program
  - Adjusted window sizes and centered them according to user's screen
- 11/27/2023: Added in skeleton code for methods used for SQLite (Chelsea)
  - Includes DB creation, table creation, and queries to register/authenticate user
- 11/27/2023: Added in user login form window (Hoi Lam)
- 11/28/2023: Added new window to display the randomly chosen fortune (Hoi Lam)

Screenshots:





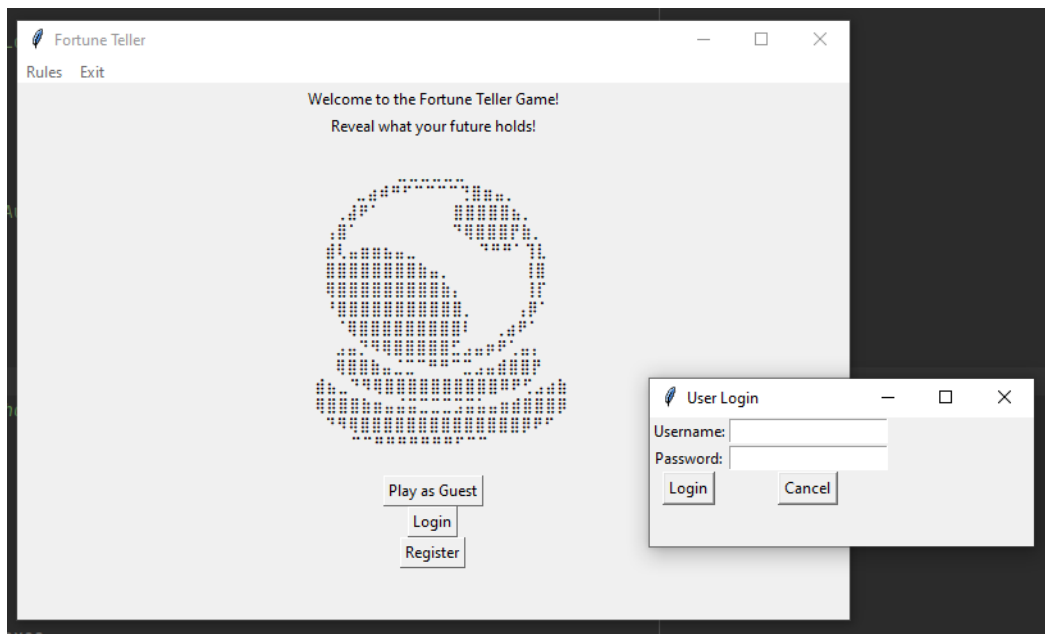

---

November 29, 2023 - December 5, 2023

### Phase 3

- 11/29/2023: Added new window to display a list of past user's fortunes (dummies data) (Hoi Lam)
  - 12/4/2023 Modified query to format SQL data output to user (Chelsea)
- 11/30/2023: Users are able to register and information is stored securely using bcrypt and SQL (Chelsea)
  - 12/4/2023 Email validation added (Valerie)
- 12/1/2023: Added new window to show confirmation message for
  - user registration and login (Hoi Lam)
  - 12/4/23 user sign out (Valerie)
- 12/1/2023 Users can log in with the matching username and password (Chelsea & Hoi Lam)
  - Created method to query db to determine if username already exists in DB (Chelsea)
  - Modified flow of "check\_all\_inputs()" and "validate\_pass()" and "sign\_up()" to return error and valid/registered variables and output error/confirmation messages to the user

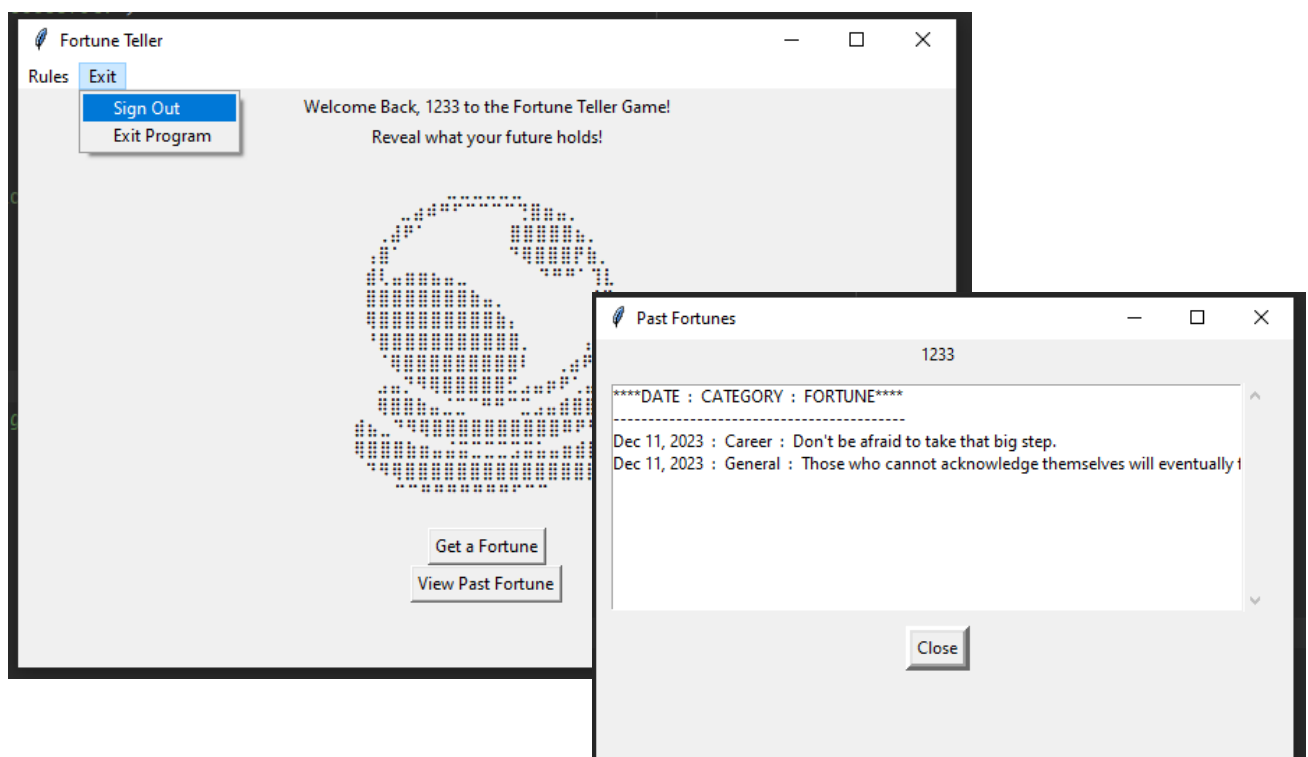
- 12/10/2023 Change has been reverted for methods to return True/False (Chelsea)
- 12/1/2023 Modified flow of GUI (Hoi Lam)
  - Added buttons to login and register in the main menu
  - Added menu bars for uniformity (Constance)
- 12/2/2023 Merge every new changes into one (Chelsea & Hoi Lam)
- 12/2/2023 Authored loghelper.db to logging files for user authentication (user\_logger) and database queries (db\_logger) (Chelsea)
  - Includes data for debug purposes
- 12/4/2023 Removed save fortune button unless user is signed in (Valerie)
- 12/4/2023 Created new window to display separate user menu after login (Valerie)
- 12/4/2023 Users can log out of their account (Valerie)
- 12/4/2023 Track user's login state and username, using global variable (Hoi Lam)
  - Display different main menu based on user's login state (login vs guest user) (Hoi Lam)
- 12/5/2023 Revised formatting per Pylint recommendations (Chelsea)
  - Revised error messages output to user to be more vague (auth failed v. no username found) per OWASP secure coding practices (improper error handling) (Chelsea)



December 6, 2023 - December 12, 2023

#### Phase 4

- 
- 12/9/2023 Fixed menu bars to always display (Constance, Valerie)
  - “Exit Program” features closes out program
  - “Sign Out” only available to users who log in
- 12/10/2023 Revised input validation methods (Chelsea)
  - Added while loops
  - Changed return functions back to true/false
  - Added separate method for username validation
  - Ensured input val methods are called before db is every queried
  - Confirmed methods work with PyUnit testing
- 12/10/2023 Continued program revisions (formatting errors, Pylint, changed some GUI windows to message boxes, removed some unnecessary Lambda func.) (Chelsea, Valerie)
- 12/10/2023 Completed writing and testing Pyunit tests to test input validation methods (Chelsea)
- 12/10/2023 Modified User and Test Guides (Chelsea)





---

## **9.0 Conclusion**

### **9.1 Lessons Learned**

- Knowing what the stakeholder fully wants implemented in the application will help with the design phase and time management of the project. Make a set of questions to ask the stakeholder ahead of the design phase. (Constance)
- Planning is key - GUI should have followed an MVC system which would have helped write the program and mitigate some issues we ran into (Chelsea)
- It is critical that team members work cohesively to maintain a single version of the program to prevent spaghetti code and multiple incompatible versions down the line that require manual merging and create confusion. (Chelsea)
- Doing group projects just using messages can cause communication issues. Adding video chats or being able to meet in person during the design phase or when making major changes could result in better collaboration. (Constance)
- Allowing the individual to fix their code before another member modifies will help the team member learn. This also allows the group member to contribute to the project. (Constance)
- Sharing research with other group members will allow for a more cohesive and uniform application as well as help mitigate minor bugs due to incompatibility. (Constance)
- Create a To-do list during the design phase to help make sure developers have their roles listed out and to keep tabs on what needs to be completed. (Constance)

### **9.2 Design Strengths**

- The GUI and program logic have remained separated to the best of our abilities, with the exception of outputting some more specific input validation errors (Chelsea)

- The GUI is user-friendly and intuitive. We have included important functionalities such as ‘Rules’ and ‘Exit’ at the top of the window, as well as including ‘Login’ and ‘Register’ in the main menu, making them easily accessible (Hoi Lam)
- Our Fortune Teller program is engaging and interactive. Users can register, create an account, and log in. Additionally, dates, fortune category, and fortune messages are recorded when a user selects to save a fortune and can be viewed at a later time through the user menu.

### 9.3 Limitations

- The GUI and database were implemented when the team did not have the ability for planning due to time constraints, thus there may still be some rough corners and security issues that require mitigating.
  - The database does not have an assigned username/password therefore any connection request is honored (Chelsea)
  - Global variables are utilized for different purposes but require mitigation (the root window in FortuneTeller.py and is\_user\_logged in from DatabaseHelper.py)
- We had limited knowledge and more research was needed on creating GUI with python and were not able to use tkinter to its full potential, resulting in a more basic and functional interface for our program such as the display of the window, buttons, and images.
- The program imports entire libraries (wildcard import), but only uses one or two items from the imported library.

### 9.4 Suggestions

- Create a new plan using an MVC system and shuffle the code around/make modifications as needed (no global root)
- Apply credentials to the database to prevent unauthorized connections

- Ideally an application (such as Flask for web servers) or some other sort of session management would be implemented to more securely monitor and log a user's authentication/authorization status.
- Modify wildcard import statements to only import objects necessary for the application
- Modify the GUI to look more modern by adding different functionalities for the buttons and window design, as well as, add real images instead of Ascii display. (Constance)

### **10.0 Links**

1. Github Repository : <https://github.com/squanchthis13/Fortune-Teller>