**A1: Hex Quiz Game Refactoring Project - Grading Rubric**

**Self-Scoring Rubric with Documentation & Consolidated Screenshots**

| Category | Criteria (with Documentation Notes) | Points | Self-Score |
| --- | --- | --- | --- |
| Frontend (30) | - getUsernameInput() & validateUsername() – username prompt, trimming, invalid vs valid input. - getValidatedChoice() – continue/quit handling. - getValidatedHexAnswer() – validation of hex input. - displayGameStats() – running score and accuracy. - displayFinalSummary() – totals, accuracy, thank-you message. **Screenshot Group A**: Program start → invalid username rejected → valid username accepted → 1–2 rounds played (correct + incorrect answers) → stats displayed → quit & final summary. | 30 |  |
| Business Logic (30) | - initializeGame() – starts game with welcome message. - playGameRound() – full round workflow: decimal generated, converted, answered, feedback shown. - convertDecToHex() – correctness tested on 0 and random values. - checkAnswer() – exact comparison; incorrect answer shows correct result. - updateScore() – increments points/correctAnswers. - finalizeGame() – flushes decimals, saves session, prints summary. Screenshot Group A already demonstrates most of this. Add **Screenshot Group B**: a session with > ARRAY\_SIZE questions to trigger decimal batch flushing. | 30 |  |
| Backend (25) | - saveQuestionHistory() – per-question logging with timestamp in questions\_history.txt. - saveSessionInfo() – session summary saved in session\_history.txt. - storeDecimal() – manages array, flushes when full. - flushDecimalsToFile() – writes batch to decimalHistory.txt. **Screenshot Group C**: File previews — show directory with questions\_history.txt, session\_history.txt, decimalHistory.txt plus their contents (2+ entries in questions file, one session summary, one decimal batch). | 25 |  |
| Code Quality & Testing (15) | - All functions implemented (no TODOs left). - Code compiles with no warnings using -Wall -Wextra. - Tested with valid/invalid inputs; files verified. **Screenshot Group D:** Compilation log + final run showing all three output files created. | 15 |  |
| TOTAL |  | 100 |  |

**EVALUATION CHECKLIST**

Before Submitting - Self-Check:

* [ ] GameState struct includes all required fields with proper types
* [ ] Constructor allocates memory, destructor deallocates memory
* [ ] Functions are organized into Frontend/Business Logic/Backend categories
* [ ] Functions with clear single responsibilities
* [ ] Username validation (3-20 characters, alphanumeric + spaces only)
* [ ] Hex input validation (0-9, a-f characters only)
* [ ] Choice validation (0 or 1 only)
* [ ] Three output files: questions\_history.txt, session\_history.txt, decimalHistory.txt
* [ ] Proper error handling for file operations
* [ ] No memory leaks (test with multiple game sessions)
* [ ] Clear, consistent code formatting and commenting
* [ ] Function categorization documented
* [ ] Sample output files included
* [ ] Screenshots of program execution provided

**Common Issues to Avoid:**

* X Mixing UI code with business logic functions
* X Memory leaks from not calling destructor
* X Functions that do multiple unrelated tasks
* X Poor input validation allowing invalid data
* X File operations without error handling
* X Inconsistent or missing documentation
* X Hard-coded values instead of using constants

**Excellence Indicators:**

* Clean separation of concerns across all functions
* Comprehensive input validation with user-friendly error messages
* Robust file management with proper error handling
* Professional code organization and documentation
* Efficient memory management with no leaks
* Thoughtful architecture that would be easy to extend

**Final Grade: \_\_\_\_\_ / 100 = \_\_\_\_\_**