**Project Proposal**

**1.1 Project Title**

**Number Guessing Game App using Python**

**1.2 Project Overview**

The Number Guessing Game App is an interactive game developed using Python. The game challenges players to guess a randomly generated number within a specified range, providing hints such as "higher" or "lower" to guide their guesses. The project will incorporate various features, including a graphical user interface (GUI), timed mode, sound effects, play levels support, animations, a leaderboard, and a Database.

**1.3 Objectives**

* Develop an interactive and user-friendly Number Guessing Game.
* Implement different game modes.
* Enhance the gaming experience with GUI, animations, and sound effects.
* Integrate a leaderboard for tracking scores.
* Implement play levels for player interaction.
* Ensure scalability and maintainability following SDLC principles.

**2. SDLC Methodology**

The SDLC model best suited for developing this app depends on its complexity and requirements. Below are two possible models:

**a) Waterfall Model (For Beginners & Small-Scale Development)**

* **Requirement Gathering**: Define game rules (e.g., range of numbers, hints).
* **Design**: Create UML diagrams, flowcharts, and UI layout.
* **Implementation**: Code the game logic, UI, and local storage.
* **Testing**: Debug and check for errors.
* **Deployment**: Publish on platforms (if needed).
* **Maintenance**: Fix bugs and improve features.

**b) Agile Model (For Continuous Enhancements)**

* Develop the game in iterative cycles (sprints).
* Receive feedback and improve continuously.
* Test after each iteration.

The **Agile Model** will be used for this project, enabling iterative development with continuous feedback and improvement.

**3. Software Development Life Cycle Phases**

**3.1 Planning**

* Define project scope and requirements.
* Identify target users.
* Allocate resources, tools, and timeline.
* Conduct risk analysis and mitigation planning.

**3.2 Requirement Analysis**

* Functional Requirements:
  + Single-player game modes.
  + Timed Mode with countdown functionality.
  + GUI-based gameplay with HTML, CSS, Bootstrap, and JavaScript.
  + Sound effects and animations.
  + Leaderboard tracking scores.
  + Real-time chat for multiplayer interactions.
* Non-Functional Requirements:
  + High performance and responsiveness.
  + Secure data handling and communication.
  + User-friendly UI/UX design.

**3.3 Design**

* **System Architecture:**
  + Client-server architecture for functionality.
  + Database integration for leaderboard.
* **UI/UX Design:**
  + Main menu with game mode selection.
  + Game screen with number input and hint display.
  + Leaderboard and play levels.
* **Technology Stack:**
  + **Frontend:** HTML for GUI
  + **Backend:** Python (Flask)
  + **Database:** PostgreSQL for storing scores and chat logs
  + **Deployment:** AWS CLI, EC2 and Elastic Beanstalk

**Algorithm for the Number Guessing Game**

1. Generate a random number within a range (e.g., 1-100).
2. Prompt the user to enter a guess.
3. Compare the guessed number with the actual number.
   * If the guess is correct → Display a success message.
   * If the guess is too high → Display "Too High" and allow another attempt.
   * If the guess is too low → Display "Too Low" and allow another attempt.
4. Track the number of attempts.
5. Store best scores using local storage.
6. Optionally, allow restarting the game.

**UML Diagram (Class Diagram & Use Case Diagram)**

**a) Use Case Diagram**

**b) Use Case Diagram**

Actors: **Player** Use Cases:

1. **Start Game** → Generate Random Number
2. **Enter Guess** → Compare with Secret Number
3. **Receive Hint** → "Too High" / "Too Low"
4. **Win/Lose Game** → Display Results
5. **Restart Game**

| **Feature** | **Description** |
| --- | --- |
| Random Number Generation | The game generates a secret number randomly. |
| User Input | Players enter their guesses. |
| Feedback System | Displays whether the guess is "Too High" or "Too Low". |
| Attempt Counter | Keeps track of the number of attempts. |
| Restart Option | Allows users to restart the game. |
| Score Saving | Stores best scores using local storage. |
| User Login/Logout |  |
| Local Storage | For user scores |
| Time Tracking |  |
| Responsive UI |  |

**3.4 Implementation**

* Develop core game logic for number guessing.
* Integrate GUI using HTML, CSS, JavaScript and Bootstrap.
* Implement timed mode using Python’s time module.
* Add sound effects using pygame.
* Store leaderboard data using PostgresSQL

**3.5 Testing**

* **Unit Testing:** Test individual components (game logic, GUI, networking).
* **Integration Testing:** Ensure seamless interaction between components.
* **System Testing:** Evaluate the complete application for performance and security.
* **User Testing:** Collect feedback from test users for usability improvements.
* **Security Testing**

**3.6 Deployment**

* Deploy the app to AWS Cloud.
* Provide installation guides and user documentation.

**3.7 Maintenance**

* Regular updates to fix bugs and improve performance.
* User feedback integration for future enhancements.
* Security updates for features.

**4. Project Timeline**

A tentative project schedule:

| **Phase** | **Duration** |
| --- | --- |
| Planning | 1 Week |
| Requirement Analysis | 2 Week |
| Design | 2 Weeks |
| Implementation | 6 Weeks |
| Testing | 3 Weeks |
| Deployment | 2 Week |
| Maintenance | Ongoing |

**5. Expected Outcomes**

* A fully functional Number Guessing Game App.
* Engaging UI with animations and sound effects.
* Secure and scalable system architecture.
* User-friendly experience with leaderboard tracking.

**6. Conclusion**

This project follows the SDLC methodology to ensure a structured and systematic development process. By incorporating engaging features, the Number Guessing Game App will provide an immersive and enjoyable user experience while demonstrating key software development principles.

**7. Possible Improvements**

**a) Advanced Features**

* **Multiplayer Mode** (Compete with friends).
* **AI Opponent** (Have an AI guess the number too).
* **Speech Recognition** (Allow voice input for guessing).
* **Mobile & Web Version** (Cross-platform support).

**b) Implementation Explanation**

To implement these improvements:

1. **Multiplayer Mode**: Use WebSockets or Firebase for real-time gameplay.
2. **AI Opponent**: Implement a binary search algorithm for efficient guessing.
3. **Speech Recognition**: Use libraries like speech\_recognition in Python or Web Speech API in JavaScript.