AI-powered Business Process Automation (BPA) application

```
#include <iostream>
#include <cstdlib>
#include <ctime>

class WorkflowEngine {
public:
    void executeWorkflow(const std::string& data) {
        // Simulate workflow execution
        std::cout << "Executing workflow with data: " << data << std::endl;
    }
};

class AIModule {
public:
    std::string analyzeData(const std::string& data) {
        // Simulate data analysis
        return "Analyzing " + data;
    }

    std::string makeDecision(const std::string& analyzedData) {
        // Simulate decision-making
        return (rand() % 2 == 0) ? "Automate" : "Human-in-the-Loop";
    }
};

class HumanInterface {
public:
    std::string getUserDecision(const std::string& data) {
        // Simulate user decision
```

```
        std::string userDecision;
        std::cout << "Human decision required. Type 'Approve' to proceed: ";
        std::cin >> userDecision;
        return userDecision;
    }
};

class BPAApplication {
private:
    WorkflowEngine workflowEngine;
    AIModule aiModule;
    HumanInterface humanInterface;

public:
    void run(const std::string& data) {
        // Simulate data-driven decision-making
        std::string analyzedData = aiModule.analyzeData(data);
        std::string decision = aiModule.makeDecision(analyzedData);

        // Simulate adaptive automation
        if (decision == "Automate") {
            workflowEngine.executeWorkflow(data);
        } else if (decision == "Human-in-the-Loop") {
            // Simulate human-in-the-loop integration
            std::string userDecision = humanInterface.getUserDecision(data);
            if (userDecision == "Approve") {
                workflowEngine.executeWorkflow(data);
            } else {
                std::cout << "Workflow halted based on human decision." << std::endl;
            }
        }
```

```
    }
};

int main() {
    // Seed for random number generation
    std::srand(static_cast<unsigned>(std::time(nullptr)));

    std::string dataInput = "Sample Data";
    BPAApplication app;
    app.run(dataInput);

    return 0;
}
```