**Bowling Game Project Structure**

**Project Files Organization**

bowling-game-project/  
│  
├── src/  
│ ├── bowling\_game.py # Original code (with bugs)  
│ ├── bowling\_game\_fixed.py # Fixed and documented version  
│ └── example\_usage.py # Usage examples and demonstrations  
│  
├── tests/  
│ ├── test\_bowling\_game.py # Comprehensive unit test suite  
│ ├── test\_results.txt # Test execution results  
│ └── coverage\_report.html # Test coverage report  
│  
├── docs/  
│ ├── bowling-test-plan.md # IEEE 829 compliant test plan  
│ ├── api\_documentation.html # Generated PythonDoc documentation  
│ ├── bowling\_rules.md # Business rules documentation  
│ └── project-summary-report.md # Final summary report  
│  
├── git\_info/  
│ ├── commit\_history.txt # Git commit log  
│ └── repository\_link.txt # Link to Git repository  
│  
├── README.md # Project overview and setup instructions  
├── requirements.txt # Python dependencies  
└── run\_tests.py # Script to execute all tests

**File Descriptions**

**Source Code (src/)**

* **bowling\_game.py**: Original implementation provided by tutor (contains bugs)
* **bowling\_game\_fixed.py**: Debugged and refactored version with comprehensive documentation
* **example\_usage.py**: Demonstrates various game scenarios and expected scores

**Tests (tests/)**

* **test\_bowling\_game.py**: Complete unit test suite covering all scenarios
* **test\_results.txt**: Output from test execution showing pass/fail status
* **coverage\_report.html**: Code coverage analysis report

**Documentation (docs/)**

* [**bowling-test-plan.md**](http://bowling-test-plan.md): IEEE 829 standard test plan
* **api\_documentation.html**: Generated PythonDoc API documentation
* **bowling\_rules.md**: Business rules and scoring explanations
* [**project-summary-report.md**](http://project-summary-report.md): Comprehensive project summary and findings

**Version Control (git\_info/)**

* **commit\_history.txt**: Git log showing all commits and messages
* **repository\_link.txt**: URL to the Git repository

**Root Files**

* [**README.md**](http://README.md): Project setup and execution instructions
* **requirements.txt**: Python package dependencies
* **run\_tests.py**: Automated test execution script

**Setup Instructions**

**1. Environment Setup**

# Create virtual environment  
python -m venv bowling\_env  
  
# Activate environment  
# Windows:  
bowling\_env\Scripts\activate  
# macOS/Linux:  
source bowling\_env/bin/activate  
  
# Install dependencies  
pip install -r requirements.txt

**2. Running Tests**

# Run all unit tests  
python -m pytest tests/ -v  
  
# Run with coverage  
python -m pytest tests/ --cov=src --cov-report=html  
  
# Or use the convenience script  
python run\_tests.py

**3. Generating Documentation**

# Generate PythonDoc documentation  
python -m pydoc -w src.bowling\_game\_fixed  
  
# Or view in browser  
python -m pydoc -b

**4. Git Repository**

# Initialize repository  
git init  
git add .  
git commit -m "Initial commit with complete project"  
  
# Push to remote repository  
git remote add origin <your-repo-url>  
git push -u origin main

**Submission Checklist**

**Required Files for Assessment:**

* [ ] Test Plan ([bowling-test-plan.md](http://bowling-test-plan.md))
* [ ] Unit Test Suite (test\_bowling\_game.py)
* [ ] Fixed Source Code (bowling\_game\_fixed.py)
* [ ] Bug Documentation (in summary report)
* [ ] Refactoring Evidence (commit history + code comments)
* [ ] PythonDoc Documentation (api\_documentation.html)
* [ ] Summary Report ([project-summary-report.md](http://project-summary-report.md))
* [ ] Git Repository Link

**Assessment Criteria Coverage:**

* [ ] **Test Planning (20 pts)**: Comprehensive IEEE 829 test plan
* [ ] **Unit Test Design (20 pts)**: Complete test suite with edge cases
* [ ] **Debugging (15 pts)**: All bugs identified and fixed with documentation
* [ ] **Refactoring (15 pts)**: Code improvements with clear justification
* [ ] **Version Control (10 pts)**: Meaningful Git commits throughout
* [ ] **Documentation (10 pts)**: Complete PythonDoc + generated docs
* [ ] **Summary Report (10 pts)**: Professional analysis and recommendations

**How to Create Project ZIP**

**For Submission:**

1. **Organize all files** according to the structure above
2. **Test everything works** by running tests and checking documentation
3. **Create ZIP file** containing entire project folder
4. **Verify ZIP contains**:
   * All source code files (original + fixed)
   * Complete test suite with results
   * Generated documentation
   * Test plan and summary report
   * Git commit history or repository link

**ZIP Creation Command:**

# Create zip file for submission  
zip -r bowling\_game\_project.zip bowling-game-project/ -x "\*.pyc" "\*\_\_pycache\_\_\*" "\*.git\*"

**Notes for Students**

**Key Success Factors:**

1. **Follow IEEE 829** for test planning structure
2. **Document all bugs found** with before/after examples
3. **Use meaningful Git commits** throughout development
4. **Generate actual PythonDoc** from your documented code
5. **Test everything thoroughly** before submission

**Common Pitfalls to Avoid:**

* Don't submit original buggy code without fixes
* Don't skip the 10th frame implementation (critical bug)
* Don't forget to generate actual documentation files
* Don't use generic commit messages
* Don't skip edge case testing

This structure ensures all assessment criteria are met and provides a professional project organization that demonstrates software engineering best practices.