|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EXPECTATIONS & ACHIEVEMENT CATEGORIES** | **Level 1** | **Level 2** | **Level 3** | **Level 4** |
| The software solution met all of the defined requirements and all test cases were executed successfully.  Knowledge [ /12] | The program meets very few of the requirements in the specifications.  Errors occur frequently.  User input needs further validation. | The program meets some of the requirements in the specification.  Some of the test cases pass.  User input needs further validation. | The program meets most of the requirements in the specification.  Most of the test cases pass.  User input needs further validation. | The program meets all of **requirements** in the specification.  All of the **test cases pass**.  User input is fully validated **robust**. |
| The source code was well designed using a modular design.  Thinking [ /8] | Program consists of very few methods.    Significant duplication of algorithms and/or data structures. | Program consists of very few methods.  Some duplication of algorithms and/or data structures. | Some methods are used, but more should be defined.  Minimal duplication of algorithms and/or data structures. | Methods are highly **cohesive** and designed for **reusability**.  **No unnecessary duplication** of **algorithms** and/or data structures. |
| The source code was written in a readable manner using industry standards.  The source code included clear internal documentation of its programming interfaces and algorithms.  Communication[ /16] | The code is poorly organized and difficult to understand.  Naming and formatting conventions for identifiers, methods, classes are not followed.  Minimal internal documentation is provided.  Minimal or no Javadoc/docstrings. | The code is somewhat organized and understand.  Naming and formatting conventions for identifiers, methods, classes are seldom followed.  Minimal internal documentation is provided.  Javadoc/docstrings is incomplete. | The code is organized and easy to understand.  Naming and formatting conventions for identifiers, methods, classes are often followed.  Internal documentation is provided, but incomplete or confusing.  Javadoc/docstrings for most methods and classes. | The code is **very well organized** and very easy to **understand**.  **Naming and formatting** conventions for identifiers, methods, classes are always followed.  Thorough **internal documentation** is provided and it **clearly explains** the design.  **Javadoc/docstrings** for each class and method |
| The source code was developed using modern software development techniques and methods  Application [ /12] | Translates a problem into a programming solution with subtasks  Only drag and drop web interface used or other forms of delete/add files  Vague and/or ambiguous description of code | Analyzes and design, decomposes a problem into program components that share data.  Only a single commit at the end, but pushed via git, not uploaded via drag/drop on web  Description of code and of approach/considerations within the code | Decomposes a program into subtasks and use parameter passing to exchange information between the subparts  Git commits and push history show that git was used in an efficient workflow  Clear description of code and explanation of approaches and considerations within the code and within the allocated time | Formulates a design strategy and decomposes a problem into program components that share data.  Git commits, push, branches, and merging history show that git was used in an efficient and consistent workflow  Structured description of code and detailed explanation of approaches and considerations within the solution in under the allocated time |