Team #_____
Project Task #2: Software Design Document (SDD)

The following is our grading rubric we will use for project task 2 – SDD .
Total points: 50

Organization (10%):

4	Organization follows the provided template. All sections present and well written. Format easy to understand. Uses good grammar and has single voice. No irrelevant data or made up content.
3	Template is followed. One small section missing. Requirements are easily understood. Lacks single voice and contains grammar issues. Template content (section descriptions) left in the document.
2	Missing some sections of the template. Layout is confusing, requirements hard to understand. Lacks single voice, many grammar issues.
1	Missing major sections or didn't follow the provided template. Lack of logical structure overall. Hard to read.

System UML Class Diagram (15%) – This a full blown UML Diagram that includes state variables and methods. All relationships are fully developed. This is your actual design of the system.

4	Diagram is completely legible. Identifies all of the classes in the system to satisfy all requirements. The multiplicities on each association (e.g. inheritance, composition, aggregation, interfaces) has no errors. Naming conventions are descriptive and clear. No irrelevant data added. All methods and state variables are accounted for.
3	Diagram is completely legible. Misses one class or system function. 1-2 multiplicities are incorrect. Naming conventions are clear in most cases. 1 irrelevant piece of data is added. Missing 1-2 methods or state variables.
2	Diagram is not always legible and is hard to read. Misses most of the classes or system functions. Multiplicities are nearly always incorrect. Naming conventions do not make sense. Two irrelevant pieces of data are added. Missing numerous methods and state variables.
1	Diagram is illegible and cannot be easily graded. Mostly incorrect and unclear. Naming conventions do not make sense. Irrelevant data is commonly added. Do not account for any of the associations. Diagram does not account for the class information (i.e. methods and state).

UML Activity Diagram for Open Party List Voting (5%)

4	Diagram is completely legible. All major activities in process are included. Process flow is logical and guards are descriptive. Activity coordination is accounted for and systems involved in the flow are correct. No irrelevant data is included. Naming conventions are clear and unambiguous.
3	Diagram is completely legible. Captures the majority of the major activities in the process flow. Process flow is mostly logical and guards are descriptive. Activity coordination is accounted for in most cases and systems involved in the flow are correct. No irrelevant systems or activities are included that are not in the writeup. Naming conventions are typically clear and descriptive.
2	Diagram is not always legible and is hard to read. Missing many of the major activities in the process flow. The flow is not always logical and the guards are missing. Activity coordinate is accounted for rarely. Irrelevant data is added frequently. Naming conventions are vague and unclear.
1	Diagram is illegible and cannot be easily graded. Flow is unclear and hard to follow. Subprocesses do not make sense and there is a lack of clarity in the diagram. Naming conventions are vague and use no descriptive language. Irrelevant data is commonly added.

UML Activity Diagram for Closed Party List Voting (5%)

4	Diagram is completely legible. All major activities in process are included. Process flow is logical and guards are descriptive. Activity coordination is accounted for and systems involved in the flow are correct. No irrelevant data is included. Naming conventions are clear and unambiguous.
3	Diagram is completely legible. Captures the majority of the major activities in the process flow. Process flow is mostly logical and guards are descriptive. Activity coordination is accounted for in most cases and systems involved in the flow are correct. No irrelevant systems or activities are included that are not in the writeup. Naming conventions are typically clear and descriptive.
2	Diagram is not always legible and is hard to read. Missing many of the major activities in the process flow. The flow is not always logical and the guards are missing. Activity coordinate is accounted for rarely. Irrelevant data is added frequently. Naming conventions are vague and unclear.
1	Diagram is illegible and cannot be easily graded. Flow is unclear and hard to follow. Subprocesses do not make sense and there is a lack of clarity in the diagram. Naming conventions are vague and use no descriptive language. Irrelevant data is commonly added.

S

Sequence Diagram for Closed Party List Voting Model (5%)

4	Diagram is completely legible. Account for all interactions with calls to objects, parameters, and return values. Interactions between components are properly modeled. Lifetime of objects is correctly captured. Alternate paths are captured. No irrelevant data (i.e. did not make up interactions). Naming conventions of calls, parameter, return values, and alternate scenarios are clear and descriptive
3	Diagram is completely legible. Missing some interactions and calls to objects, parameters, and return values. One interaction between objects is improperly modeled. Lifetime of objects is generally correct. Not all calls, parameters, return values or alternative scenarios have description names. Some alternate paths are captured. One irrelevant interaction is added.
2	Diagram is not always legible and is hard to read. Missing many interactions and calls to objects, parameters, and return values. More than one interaction between objects is improperly modeled. Lifetime of objects is rarely modeled properly. Missing most alternate paths. Few all calls, parameters, return values or alternative scenarios have description names. Two irrelevant interactions are added.
1	Diagram is illegible and cannot be easily graded. Missing nearly all interactions and calls to objects, parameters, and return values. Lifetime of objects is incorrect. Missing most calls, parameters, return values or alternative scenarios. No alternate paths indicated. Do not use meaningful naming conventions. Many irrelevant interactions are included.

Software Design Document Requirements (60%)

4	All major and minor system functionality requirements have designs to satisfy them. Accounts for error cases. All sections are sufficiently complete. No contradictory information. Designs are sensible and easy to understand. Rationale is clear.
3	All major system functionality requirements have designs to satisfy them; majority of minor system functionality satisfied. Some error cases accounted for. Most sections are sufficiently complete. Designs are mostly sensible. Rationale is mostly clear
2	Most major and few minor system functionality requirements have designs to satisfy them. Error cases missing. Some sections missing most detail. Design is illogical or counterproductive. Rationale is hard to understand or misguided.
1	Most system functionality requirements have no designs. Generally, design doesn't satisfy requirements of project. Rationale is confusing or contradictory or flat out incorrect. Many sections completely missing.

((_____* .10) + (___* .15) + (___* .05) + (___* .05) + (___*.05) +(___* .60))/4 * 50 =_____