Project Proposal Template

Student Name(s): Your name(s)
Proposed System Under Test (SUT): Name of the system under test
Link to SUT Source Code: Link to the system under test
SUT Size: Number of lines of code in the system under test

SUT Description
What is the SUT, and what does it do?

State of SUT
How old is the SUT? Is it being actively developed? Is there documentation available? Is the source code messy?

Attributes
From an ACC perspective, what attributes does the SUT have? Name all relevant attributes, even if your testing doesn't intersect with them. Generally you'll have 3-7, never more than 12.

Components
From an ACC perspective, what components does the SUT have? Name all relevant components, even if your testing doesn't intersect with them. Generally you'll have several; rarely more than 12, never more than 20.

Capabilities:
From an ACC perspective, what capabilities does the SUT have? Each capability should clearly list which attribute and component the capability intersects with. In general, there may be hundreds of capabilities. In this case, list **only** those capabilities you're planning to test with basic industry-standard approaches (e.g., unit/integration tests). You must list **at least one** capability. The fewer capabilities you list, the more I expect from advanced testing.

Capabilities which intersect more than one attribute/component should be split into multiple capabilities which each intersect with one attribute and component. Capabilities which intersect with neither indicate one of the following problems:

- An attribute is missing (simply add it)
- A component is missing (simply add it)
- The SUT has a capability which is irrelevant to its attributes or components (it happens in real software!) If this is the case, additionally include why it doesn't relate to your listed attributes or components, why this seemingly-irrelevant feature is in the SUT, and why it's still worth testing.

Capabilities Count:
Distilling the attributes, components, and capabilities above, distill the information into a table putting capabilities horizontally and attributes vertically. For each cell, show the

number of capabilities you have put in each component/capability intersection. Do not count any oddball capabilities which don't fit neatly into attribute/component pairs.

Basic Testing:
Using industry-standard techniques (first four weeks of class), how are you planning to test the capabilities you previously mentioned, and what does this involve? In most cases, the answer involves writing unit and integration tests. However, **how difficult do you anticipate this being**?

If the codebase is already heavily tested, this likely won't be difficult; the code is likely structured in a way that adding new tests is easy. However, if the code isn't well-tested (or not tested at all!), you will likely need to refactor the code to isolate components worth testing. For this class, I consider this refactoring **a legitimate V&V exercise**, since good V&V often requires the SUT to be structured a certain way. In other words, your basic testing will be considered to be a bigger contribution if you need to heavily modify the code to make this work.

Advanced V&V:
How are you planning to apply more advanced V&V techniques (weeks 5+, particularly 5-9) to the SUT? These techniques do not necessarily have to be taught in the class. This section does not have to directly relate to your ACC test plan, though you may be able to reuse some work if it does.

If your basic testing component is a particularly massive contribution, this section may be empty (i.e., it's ok if you're not planning to apply advanced V&V techniques). However, this only applies if there truly is a **massive** amount of work in the basic testing component. It is expected to be less work to undertake a moderate basic testing effort and a small advanced V&V effort than to choose a massive basic testing effort and no advanced V&V.

Choosing advanced V&V techniques discussed later in the course will be met with higher value. That said, you should **not** wait until late in the course to deploy these techniques; I can give you information to start with ahead of time. On a related note, the V&V techniques after week 9 tend to have a very high learning curve, and it's likely infeasible to apply them completely to your SUT (e.g., it is likely unreasonable to verify your entire SUT correct).

---END TEMPLATE AND BEGIN EXTRA INFORMATION / ADVICE---

- Depending on your proposal, I may ask you to add content or scale back. It's better to propose more and have me tell you to scale back than it is to propose less and have me add something to your plate, as you'll have more control over the process.
- At the time of the proposal, the proposal is **NOT** a hard commitment, but a plan. I'm not expecting you to have complete knowledge of the SUT or testing techniques at proposal time, so you cannot make perfect decisions at proposal time. **This is ok**; the proposal is just to get you started. If you later find out that the proposal is asking for too much, **talk to me ASAP**, and we can revise it.

- On the other hand, if you discover your proposal asks for too little, no further action is required on your end. You may, however, request an addition to your proposal for bonus points. Worse case scenario, if you request an addition and only meet your original proposal, you will not be penalized.
- While the proposal is initially not a hard commitment, it becomes more of a commitment as the course progresses. Initially, I'm likely to allow you to scale back without penalty. As time progresses, I may attach a penalty to scaling back (but not *requesting* to scale back), or I may not allow you to scale back at all. Requests to scale back after week 12 will likely be met with severe penalty, particularly if the V&V approaches you choose do not involve material later in the course.
- Students may work in pairs, and submit the same proposal. However, more will be expected of such proposals.
- Different students/groups of students may work on the *same SUT*, as long as their efforts are distinct from each other (i.e., they don't write exactly the same tests, or employ the same advanced V&V technique). It is expected that this will only be applicable for very large SUTs (likely hundreds of thousands of lines long).