## **Executive Summary**

* Strategic Motion Video is a [insert project classification here startup/research project/educational company?] offering educational video software for corporations to train their [employees/customers?]
* SMV provides an interface for students to pause a training video to ask a question they do not feel is clear from the video
* SMV wishes to identify the best possible answer to that question using Natural Language Processing techniques if the information is covered in their database of answers to possible questions
* So far, we have...

The Executive Summary should provide a one-page summary that includes a brief description of the problem given to the team as well as the work accomplished by the team *at the time of report submission*. For a reader who is just interested in a quick account of the problem you are tackling and what you’ve managed to accomplish thus far, this should provide a rough idea of what the team has done. For a reader who will go on to read the remainder of the document, this should provide an overview of what to expect in the following sections.

The Executive Summary should stand alone (insert a page break at end of the section – you do not need to do this for any other section of the IPR).

## **Project Description**

The next five sub-sections describe the project, how it is important to the sponsor, and also how the team has gone about creating a solution. They also discuss any issues that might become impediments to the project’s success.

### **Sponsor Background**

In your own words, give a brief description of your sponsor’s location, business, and technical motivation for the project. Describe in greater detail the part of the business that is related to the project in order to provide the necessary context for your project.

In a section blocked off from the main paragraph, provide the name, title, address, phone, and e-mail of your sponsor contact(s).

### **Problem Statement**

Your team’s starting point is the problem statement or paragraph furnished by your sponsor that was presented to you on the first day of class. Your task in this section of the report is to elaborate on the problem statement by presenting the need addressed by the proposed software. If applicable, describe the existing system that will be extended or improved by this project; describe its basic operation, focusing on its perceived deficiencies. Otherwise, describe the need that a new system will satisfy.

Be specific; try not to only use words such as “better”, “improve”, or “inefficient”, for example. Try to identify a metric that will quantify how the new system might be better, improved or more efficient.

### **Project Goals & Benefits**

Identify and describe the major goals of the project. In general terms, what will you deliver to the sponsor? Describe the system to be created, or describe the improvements targeted for the current system. Avoid general qualifications, such as “better,” in your goal statement. Instead, explain that the proposed software will be faster, more accurate, more user-friendly, etc. than the current system.

Describe the benefits to the sponsor, to system users within the organization or from the organization’s clientele, and to the community at large. Why would the sponsor want a software system of the type you are developing?

This section is a follow-up to the problem statement. You must, therefore, check that any goal or benefit you introduce in this section addresses a problem discussed in the problem section. If it does not, revise your problem statement.

In this section, avoid uncertain language: the project “should,” “probably” will, “most likely” will, etc. Assume, and express, confidence.

### **Development Methodology**

Briefly describe the development methodology the team is using for the project. For example, if the team uses an iterative process, be sure to indicate the iteration length and number of iterations. Indicate how often the team is meeting with sponsors.

The selection of a development methodology should be done in consultation with the team’s sponsor and their technical advisor.

### **Challenges & Resolutions**

At the start of a software development project, the development team should consider technical challenges and legal issues (defined below). Report significant concerns related to these issues, and, as the project matures, **record how these issues are resolved.**

***Technical Challenges***

Discuss challenges due to technologies, domain background, or any other issues that may affect the team’s ability to achieve an acceptable solution.

Technical feasibility is frequently the most difficult area to assess at the initial stages of the product engineering process. Because project objectives, requirements, performance, etc. are somewhat hazy, anything seems possible if the “right” assumptions are made (or *impossible* if worst case assumptions are made). It is essential that the process of initial project analysis and the ensuing refinement of scope be conducted in parallel with an assessment of technical feasibility.

The considerations that are normally associated with technical feasibility include:

* Development risk. Can the system element be designed so that necessary functions and performance are achieved within the constraints uncovered during analysis?
* Technology. Has relevant technology progressed to a state that will support the system? Does the team have, or can they acquire, the skills and knowledge necessary to build the system?
* Resource availability. Are other necessary resources (hardware and software) available to build the system?

***Legal Issues***

Legal issues span a broad range of topics including contracts, liability, copyright or license infringement, and a myriad of other concerns frequently unfamiliar to technical staff. A detailed discussion of legal issues related to software design and development is beyond the scope of this document. However, students should be aware of guidelines published by NC State University (see section 8 at [http://research.ncsu.edu/ott/files/2011/12/Inventors-Guide-1.pdf)](http://research.ncsu.edu/ott/files/2011/12/Inventors-Guide-1.pdf)

In some cases, company sponsors of Senior Design projects will ask students to assign their intellectual property rights to the company or to sign nondisclosure agreements. Students may confer with Student Legal Services, for free, so they can better understand their rights and legal liabilities.

### **Resources Needed**

In both the IPR and FPR, this section should include all software and hardware resources needed to complete the project, along with a brief explanation or justification for each item. Each resource should include a status indicating whether or not it has been obtained.

This section should also include a description of *External Dependencies and Interfaces*:

* Describe, in detail, all connections to external systems.
* Describe any input, processing, or output requirements involving other applications upon which this project is dependent.
* Describe any software requirements, hardware requirements, additional cables or phone lines, special forms, and so forth that must coincide with the installation of this project.
* Describe any special support requirements from other areas, such as computer operations, database support, dependency managers, or library control, which this project will require.

## **Requirements**

The Requirements Definition portion of your documentation describes the capabilities that the proposed system must have or what it must be able to do. Outline the requirements the system must meet in order to accomplish its overall goals.

In general, teams can expect to craft their own requirements based on the initial project description and communication with the sponsor. The requirements section of the report is an essential tool for communicating with the sponsor and making sure the team is solving the right problems. As early as possible in the semester, teams should give their sponsor an opportunity to review the requirements and fine-tune the direction of the project as necessary.

Document your requirements in a way that is consistent with your experience from CSC 326, Software Engineering. The requirements section should include the following sub-sections.

### **Overall View**

Give the reader a quick but overall look at the project from the user perspective. If there is a current system in place, provide details on it, also noting what changes the team will implement to this system and why. This section should provide a “big-picture” view of what the application will do, as well as explain the application tasks to be performed and how this meets the needs of the company. (Pressman, *Software Engineering, A Practitioner’s Approach*, McGraw-Hill, 1997, pp. 253-255.)

### **Formal Requirements**

This section should be organized into *Functional Requirements, Non-Functional Requirements,* and *Constraints*.

Functional and non-functional requirements should be presented in an outline format. It is important to identify users, user tasks, and all details of what the proposed system must do. Be specific in the requirements, producing several fine-grained items rather than a few vague items. Giving each specific requirement a unique identifier will help the team to justify design decisions, test individual aspects of functionality, and schedule team activities.

[⬇︎ Sample functional requirements](https://sdc.csc.ncsu.edu/coursepack/wp-content/uploads/2017/02/Sample-Functional-Requirements.docx)

[⬇︎ Sample use case descriptions](https://sdc.csc.ncsu.edu/coursepack/wp-content/uploads/2015/01/SampleUseCaseDescriptions.doc)

## **Design**

The design section describes the structure and operation of the software product. This may include descriptions of relevant classes and components as well as GUI mock-ups, descriptions of file formats, database schemas, and data structures.

It may be helpful to illustrate aspects of the design using relevant UML notation, such as Class or Sequence Diagrams. Where the project uses a third-party framework or API, it may be helpful to briefly describe the interface between the project and the software on which it depends.

This overall design is the basis for the implementation of the project. It should provide the following information:

* Description of High-Level Design / System Architecture / Gross Program Logic
  + Overview of Modules (e.g., Classes) & Relationships Between Them
  + Sequence Diagram (e.g., UML Object Message Diagram, Flow Chart)
* Description of Low-Level Design (In Tabular, Narrative, or Pseudo-Code Form)
* For Each Module Describe:
  + Purpose
  + Overall Behavior
* If the project has a database, include Entity Relationship Diagrams and/or Table Design
* If the project has GUIs, show prototype(s) and evolution of prototype based on feedback received

In documenting the design, teams should take advantage of the work done to craft the project requirements. Where possible, use requirements to explain or justify various aspects of the design. In documenting the design, teams should consider the question: *what would a new developer need to understand in order to make meaningful contributions to the implementation?*

The final report includes a developer’s guide as a separate document (see [below](https://sdc.csc.ncsu.edu/coursepack/written-reports/#developers-guide)). Frequently, questions are asked about which elements of design should be documented in the design section and which should be documented in the developer’s guide. Design elements that offer natural avenues for extension should be documented in the developer’s guide, as should parts of the design that are expected to require modification for deployment at the sponsor’s site. Elements of the design and implementation that the sponsor may need to change (e.g., to configure or extend the product) should be documented in the developer’s guide. All other design features should be documented here.

[⬇︎ Sample project storyboard](https://sdc.csc.ncsu.edu/coursepack/wp-content/uploads/2015/01/SampleStoryboarding.doc)

## **Implementation**

In the Interim Project Report, this section should describe the current status of the code implementation at the time of report submission. In the Final Report, this section should describe the final state of the implementation.

Define your iterations in this section [e.g., “Iteration 1 is defined by the following features: x, y, z (Requirements 1.1, 1.2 and 1.3)”]. This section should also describe major source code components, the project folder structure, and project configurations and settings. If applicable, it should include examples of particular algorithms chosen for implementation. Details of how source code directories are organized and how design elements are mapped to various source and data files should be presented in the [Developer’s Guide](https://sdc.csc.ncsu.edu/coursepack/written-reports/#developers-guide).

## **Test Plan & Results**

Teams will normally plan for acceptance testing of the entire software product and unit testing of individual code components. At the completion of the project, teams will have a mature plan for both acceptance and unit testing. In the final report, these sections will not only list individual test cases but will also report on the results of each test. Acceptance and unit testing each make up half of the team’s testing grade.

[⬇︎ Sample Testing Report](https://sdc.csc.ncsu.edu/coursepack/wp-content/uploads/2015/01/SampleSoftwareTestingReport.doc)

### **Unit Testing**

Each team must include a unit test suite of automated white box unit tests for *every*component of the project (that is, across all operational environments and languages). Most coverage tools provide a way to obtain a printed coverage report after execution of the unit test suite. Each team must include a copy of their current coverage report in each progress report and the final project report.Each team should list the testing tool used to perform unit testing and to collect coverage statistics, as well as any classes which were not well covered and the reasons why these units were not tested.

Each team should list the testing tool used to perform unit testing and to collect coverage statistics, as well as any classes which were not well covered and the reasons why these units were not tested. The unit test suite should reach 70% statement coverage across all units that are deemed testable. Exempt units include:

* GUIs
* Utility – for example: used to prepare the system for manual testing or deployment
* System libraries
* Third-party – for example: contained in an open source framework
* Possibly others – these should be discussed with your technical advisor

If there is no tool support for a language or environment, an ad-hoc scripting approach can be used. That is, students will be required to write standalone testing components which cover their code units, and more grading emphasis will be put on integration testing. In this case, the coverage criteria requirement does not hold. If scripting is infeasible, the team can choose to include more acceptance test cases which test the boundary conditions or exceptional cases of their system. The team’s technical advisor will determine whether the team has sufficiently compensated for the lack of unit testing.

For more review on Requirements Development and Testing, please refer to <http://www.cs.umd.edu/~mvz/cmsc435-s09/pdf/Williams-draft-book.pdf>

## **Task Plan (IPR only)**

The purpose of the Project Task Plan is to provide a working list of all development activities. This will become a commitment of resources and scheduled work items and tasks against which project tracking will be measured. Create the project task plan as follows:

* Identify the scope and size of the project. Identify major tasks and break them down into subtasks. Assign responsibilities to team member(s) and list them next to assigned tasks.
* Include dates for all tasks and subtasks determined for the project. Due dates should be included as well as a status column. Include related coursework such as presentations and sponsor meetings.
* List the name, email, phone number, and team role(s) of each team member.

[⬇︎ Sample Task Plan](https://sdc.csc.ncsu.edu/coursepack/wp-content/uploads/2015/01/SampleProjectTaskPlan.doc)

## **Suggestions for Future Teams**

While completing the project, your team will certainly have learned some things that would be valuable to any developers wishing to continue your work. In this section, document any development tips learned during the project, opportunities for making the project more functional or for redesigning to improve maintainability. Include feature suggestions for future implementations of the software.

# **Supplemental Guides**

These separate documents are to be submitted along with the Final Report. [Guidelines for the Developer’s Guide, Installation Guide, and User’s Guide are provided separately](https://sdc.csc.ncsu.edu/coursepack/written-reports/supplemental-guides/).