

# **Software Design Document for VR Training**

**Version 1.01**

Prepared by Cameron Cheng, Fernando Torres, Denise Tabilas, Martin Castorena,  
Manuel Guillen, Jaiden Holcomb, Seng Hei Lei, Matthew Mendoza, Kevin Truong  
and Han Cao

Southern California Edison

11/21/22

Table of Contents.....	<pg #>
Revision History.....	<pg #>
1. Introduction.....	<pg #>
1.1. Purpose.....	<pg #>
1.2. Document Conventions.....	<pg #>
1.3. Intended Audience and Reading Suggestions.....	<pg #>
1.4. System Overview.....	<pg #>
2. Design Considerations.....	<pg #>
2.1. Assumptions and dependencies.....	<pg #>
2.2. General Constraints.....	<pg #>
2.3. Goals and Guidelines.....	<pg #>
2.4. Development Methods.....	<pg #>
3. Architectural Strategies.....	<pg #>
4. System Architecture.....	<pg #>
4.1. ....	<pg #>
4.2. ....	<pg #>
5. Policies and Tactics.....	<pg #>
5.1. Specific Products Used.....	<pg #>
5.2. Requirements traceability.....	<pg #>
5.3. Testing the software.....	<pg #>
5.4. Engineering trade-offs.....	<pg #>
5.5. Guidelines and conventions.....	<pg #>
5.6. Protocols.....	<pg #>
5.7. Maintaining the software.....	<pg #>
5.8. Interfaces.....	<pg #>
5.9. System's deliverables.....	<pg #>
5.10. Abstraction.....	<pg #>
6. Detailed System Design.....	<pg #>
6.x Name of Module.....	<pg #>
6.x.1 Responsibilities.....	<pg #>
6.x.2 Constraints.....	<pg #>
6.x.3 Composition.....	<pg #>
6.x.4 Uses/Interactions.....	<pg #>
6.x.5 Resources.....	<pg #>
6.x.6 Interface/Exports.....	<pg #>
7. Detailed Lower level Component Design	
7.x Name of Class or File.....	<pg #>
7.x.1 Classification.....	<pg #>
7.x.2 Processing Narrative(PSPEC).....	<pg #>
7.x.3 Interface Description.....	<pg #>
7.x.4 Processing Detail.....	<pg #>
7.x.4.1 Design Class Hierarchy.....	<pg #>
7.x.4.2 Restrictions/Limitations.....	<pg #>
7.x.4.3 Performance Issues.....	<pg #>
7.x.4.4 Design Constraints.....	<pg #>
7.x.4.5 Processing Detail For Each Operation.....	<pg #>

8.	User Interface	
8.1.	Overview of User Interface.....	<pg #>
8.2.	Screen Frameworks or Images.....	<pg #>
8.3.	User Interface Flow Model.....	<pg #>
9.	N/A	
10.	Requirements Validation and Verification.....	<pg #>
11.	Glossary.....	<pg #>
12.	References.....	<pg #>

# Revision History

Name	Date	Reason For Changes	Version
Martin Castorena	11-21-2022	Work on 2.1,2.2,5.1,5.2,5.3	1.0
Fernando Torres	12-8-22	working on all of part 1	1.1
Jaiden Holcomb	12-9-2022	Implementing User Interface Section (Section 8.1 - 8.3)	1.1

<Add rows as necessary when the document is revised. This document should be consistently updated and maintained throughout your project. If ANY requirements are changed, added, removed, etc., immediately revise your document.>

# 1. Introduction

## 1.1 Purpose

The purpose of this project is to develop a training application in virtual reality for Southern California Edison. By training in VR, this ensures the safety of the trainees while learning important skills in doing overhead and underground tasks.

Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.

## 1.2 Document Conventions

**we went to chino hills edison in november for guidance**

Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.

## 1.3 Intended Audience and Reading Suggestions

**Edison trainee or outsource trainee**

Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

## 1.4 System Overview

**unity**

Provide a general description of the software system including its functionality and matters related to the overall system and its design (perhaps including a discussion of the basic design approach or organization)

## 2. Design Considerations

### 2.1 Assumptions and Dependencies

- User has read the VR manual
- User contain a Meta Quest 2
- User knows how to work basic tools, and knows how to operate a utility truck bucket

### 2.2 General Constraints

#### User Experience

- Consumer will be placed into one scene not many different scenes
- Utility truck bucket cannot be moved
- User cannot interact with environment freely, since products are manually placed

#### System Constraints

- Meta Quest 2 is the only headset usable for project

### 2.3 Goals and Guidelines

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own subsection if you wish. Such goals might be:

- The KISS principle ("Keep it simple stupid!")
- The Software has a mandatory delivery date that must be met (end of the cd3337 class)
- Emphasis on speed versus memory use
- The product should work, look, or "feel" like an existing product

## 2.4 Development Methods

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

These would be things such as the 'Water Fall Development' methods, 'Agile Development', 'Unplanned Mad Scramble Development', or other development models and variations. Describe how these were applied in the case of your project.

### 3. Architectural Strategies

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

- **Use of a particular type of product (programming language, database, library, etc. ...)**
  - **c plus plus**
- **Reuse of existing software components to implement various parts/features of the system**
- **Future plans for extending or enhancing the software**
- **User interface paradigms (or system input and output models)**
- **Hardware and/or software interface paradigms**
- **Error detection and recovery**
- **Memory management policies**
- **External databases and/or data storage management and persistence**
- **Distributed data or control over a network**
- **Generalized approaches to control**
- **Concurrency and synchronization**
- **Communication mechanisms**
- **Management of other resources**

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

## 4. System Architecture

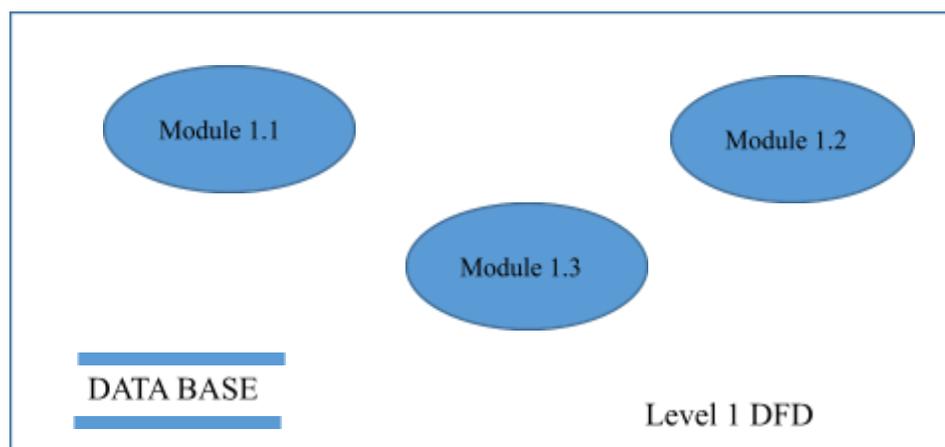
This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.



This is where the level 0 DFD will probably work best.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it).

Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.



Level 1 Data Flow Diagrams (DFD) and Control Flow Diagrams (CFD) should probably go here.

Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

## 5. Policies and Tactics

### 5.1 Choice of which specific products used

IDE: Unity

Blender

### 5.2 Plans for ensuring requirements traceability

Requirements were discussed at the beginning of the project. If requirements aren't clear, the team goes back to SRS documentation to see what requirements have not been met. If a requirement is not addressed in SRS, then it will be brought up during the meeting with SCE.

### 5.3 Plans for testing the software

#### 5.3.1 Testing the features of the modules

- Sessions

Testing user experience as we are developing the experience

#### 5.3.2 Engineering trade-offs

- Sessions don't support different VR headsets

#### 5.3.3 Coding guidelines and conventions

- Coding Guidelines
  - Code must have comments explaining what the purpose of the code
  - Comments at the head of a script explaining the purpose of the script

#### 5.3.4 The protocol of one or more subsystems, modules, or subroutines

- N/A

#### 5.3.5 The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

- N/A

#### 5.3.6 Plans for maintaining the software

#### 5.3.7 Interfaces for end-users, software, hardware, and communications

#### 5.3.8 Hierarchical organization of the source code into its physical components (files and directories)

#### 5.3.9 How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

#### 5.3.10 Tactics such as abstracting out a generic DatabaseInterface class

## 6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD's will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

### 6.1 Main Menu Module (MMM)

#### 6.1.1 Responsibilities

6.1.1.1 Allow the user to select the environment to train in.

6.1.1.2 Environment available will be overhead or underground

#### 6.1.2 Constraints

6.1.2 Language will be in English

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

#### 6.1.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

#### 6.x.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass's, and metaclasses.

### 6.x Name of Component (Module)

### **6.x.1 Responsibilities**

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

### **6.x.2 Constraints**

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

### **6.x.3 Composition**

A description of the use and meaning of the subcomponents that are a part of this component.

### **6.x.4 Uses/Interactions**

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass's, and metaclasses.

### **6.x.5 Resources**

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

### **6.x.6 Interface/Exports**

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

## **7. Detailed Lower level Component Design**

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”.) Note that there isn't necessarily a one-to-one correspondence between packages and components.

### **7.x Name of Class or File**

#### **7.x.1 Classification**

The kind of component, such as a subsystem, class, package, function, file, etc.

#### **7.x.2 Processing Narrative (PSPEC)**

A process specification (PSPEC) can be used to specify the processing details

#### **7.x.3 Interface Description**

#### **7.x.4 Processing Detail**

##### **7.x.4.1 Design Class Hierarchy**

Class inheritance: parent or child classes.

##### **7.x.4.2 Restrictions/Limitations**

##### **7.x.4.3 Performance Issues**

##### **7.x.4.4 Design Constraints**

##### **7.x.4.5 Processing Detail For Each Operation**

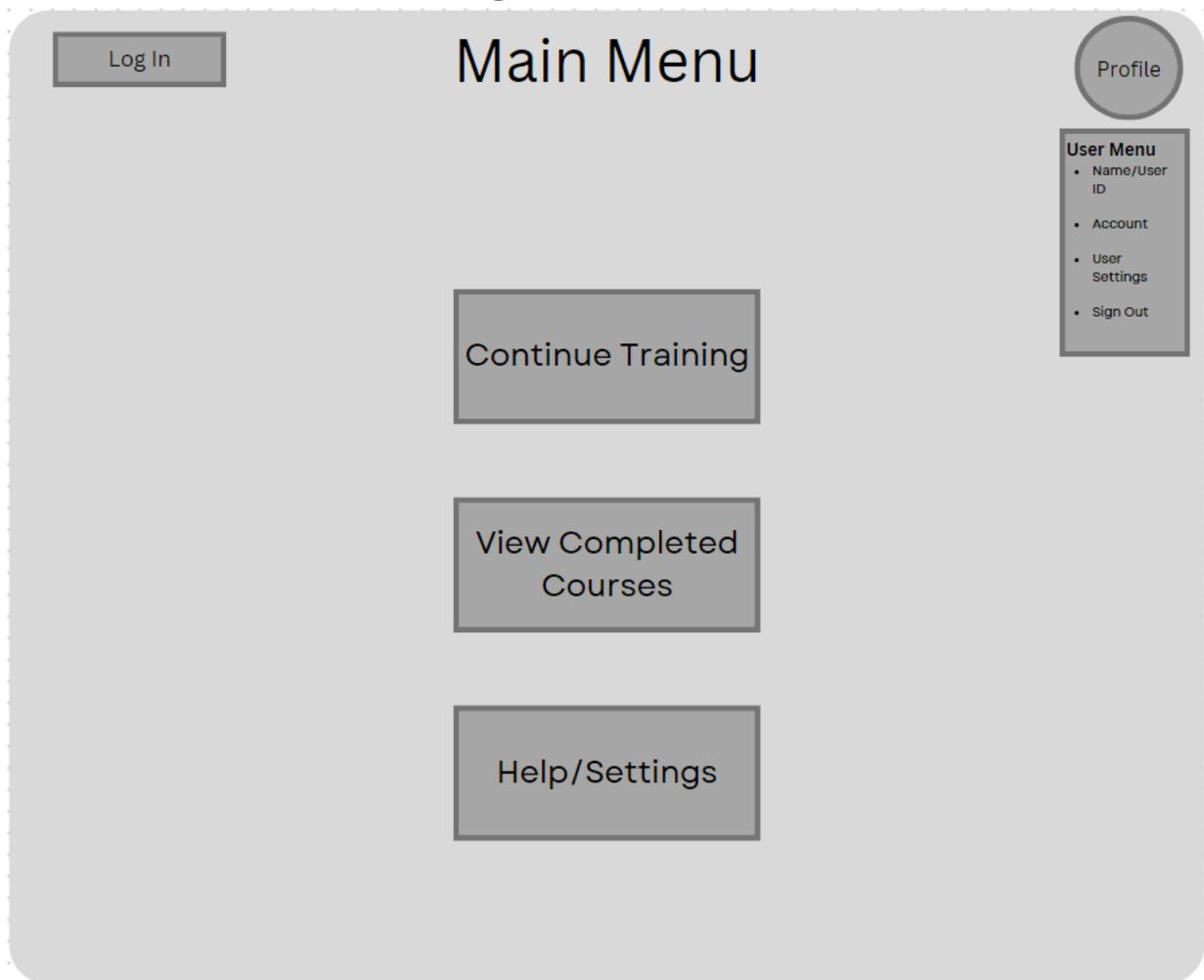
## 8. User Interface

### 8.1 Overview of User Interface

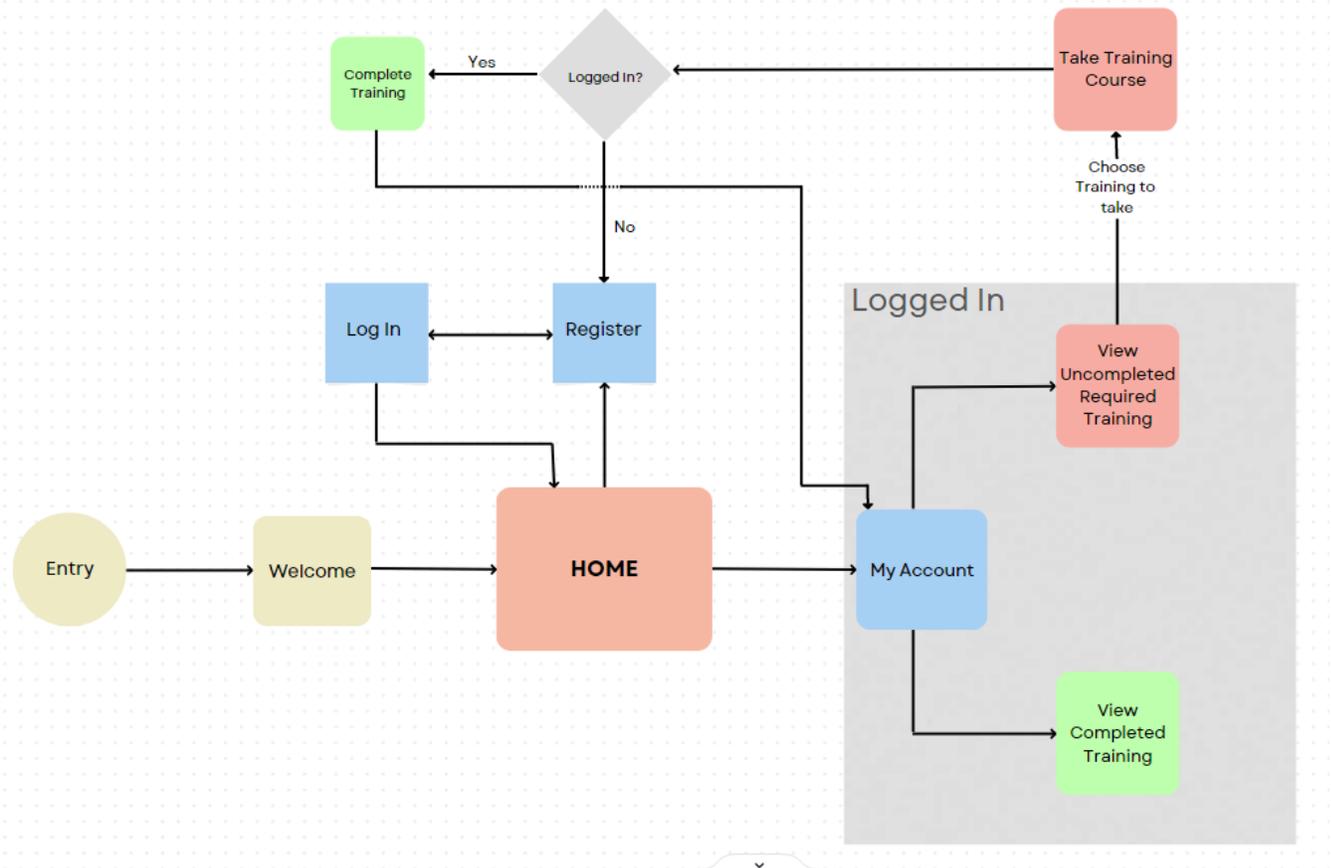
The application is designed to help new Edison employees to use VR to train and prepare for field work and machine operation. Therefore, it's important that all of our UI is simple and accessible to everyone. Our main menu will have the options to complete training, in which users can view which machinery demo they want to try. The next option will be to view completed training courses, in which users will be able to view all training simulations that have been done and have the option to re-complete them. The last major option is Help and Settings, where users can adjust various settings within the program, such as volume and language.

The user must Log in using their Edison Employee Account in order to start training courses. The ability to Log in will be in the top right hand corner. Furthermore, the user will be redirected to the Login page if any of the three initial options are selected while there is no current user account logged into

### 8.2 Screen Frameworks or Images



### 8.3 User Interface Flow Model



## 10. Requirements Validation and Verification

Create a table that lists each of the requirements that were specified in the SRS document for this software.

For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.

For each entry describe the method for testing that the requirement has been met.

Requirements Validation	Date	Description	Check
Completeness checks			<input type="checkbox"/>
Consistency checks			<input type="checkbox"/>
Validity checks			<input type="checkbox"/>
Realism checks			<input type="checkbox"/>
Ambiguity checks			<input type="checkbox"/>
Verifiability			<input type="checkbox"/>

### Requirements Verification:

1. Product and product component requirements
  - a.
2. Standards
  - a.
3. Organizational policies
  - a.
4. Test type
  - a.
5. Test parameters
  - a.

## 11. Glossary

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document. This information may be listed here or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

## 12. References

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Brad Appleton <brad@bradapp.net> <http://www.bradapp.net>

[https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate\\_Fall08.doc](https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc)