|  |
| --- |
| [Term Paper Title] |
|  |

*This report template is based on the work of* [*Ivan Marsic*](http://www.ece.rutgers.edu/~marsic/) *at Rutgers, The State University of New Jersey.*

Table of Contents

[Customer Statement of Requirements 1](#_Toc432855741)

[Problem Statement 1](#_Toc432855742)

[Glossary of Terms 1](#_Toc432855743)

[System Requirements 2](#_Toc432855744)

[Enumerated Functional Requirements 2](#_Toc432855745)

[Enumerated Nonfunctional Requirements 2](#_Toc432855746)

[On-Screen Appearance Requirements 2](#_Toc432855747)

[Project Management and References 3](#_Toc432855748)

[Merging the Contributions from Individual Team Members 3](#_Toc432855749)

[Project Coordination and Progress Report 3](#_Toc432855750)

[History of Work 3](#_Toc432855751)

[Breakdown of Responsibilities 3](#_Toc432855752)

[Functional Requirements Specification 4](#_Toc432855753)

[Stakeholders 4](#_Toc432855754)

[Actors and Goals 4](#_Toc432855755)

[Use Cases 4](#_Toc432855756)

[Casual Description 4](#_Toc432855757)

[Use Case Diagram 5](#_Toc432855758)

[Traceability Matrix 5](#_Toc432855759)

[Fully-Dressed Description 5](#_Toc432855760)

[System Sequence Diagrams 5](#_Toc432855761)

[User Interface Specification 5](#_Toc432855762)

[Preliminary Design 6](#_Toc432855763)

[User Effort Estimation 6](#_Toc432855764)

[Domain Analysis 6](#_Toc432855765)

[Domain Model 6](#_Toc432855766)

[System Operation Contracts 7](#_Toc432855767)

[Mathematical Model 7](#_Toc432855768)

[Interaction Diagrams 7](#_Toc432855769)

[Class Diagram and Interface Specification 7](#_Toc432855770)

[Class Diagram 8](#_Toc432855771)

[Data Types and Operation Signatures 8](#_Toc432855772)

[Traceability Matrix 8](#_Toc432855773)

[Object Constraint Language (OCL) Contracts 9](#_Toc432855774)

[System Architecture and System Design 9](#_Toc432855775)

[Architectural Styles 9](#_Toc432855776)

[Identifying Subsystems 9](#_Toc432855777)

[Mapping Subsystems to Hardware 9](#_Toc432855778)

[Persistent Data Storage 9](#_Toc432855779)

[Network Protocol 10](#_Toc432855780)

[Global Control Flow 10](#_Toc432855781)

[Hardware Requirements 11](#_Toc432855782)

[Algorithms and Data Structures 11](#_Toc432855783)

[Algorithms 11](#_Toc432855784)

[Data Structures 11](#_Toc432855785)

[User Interface Design and Implementation 12](#_Toc432855786)

[Design of Tests 12](#_Toc432855787)

[References 13](#_Toc432855788)

[Reflective Essays 13](#_Toc432855789)

[Team Member: <Name> 13](#_Toc432855790)

[Team Member: <Name> 13](#_Toc432855791)

[Team Member: <Name> 13](#_Toc432855792)

[Team Member: <Name> 13](#_Toc432855793)

[Team Member: <Name> 13](#_Toc432855794)

[Team Member: <Name> 13](#_Toc432855795)

[Team Member: <Name> 13](#_Toc432855796)

# Customer Statement of Requirements

## Problem Statement

A minimum 3-page high-level narrative about your project. The narrative should not be written from the developer’s perspective, describing the features of the planned system. Rather, put yourself into a customer’s role, and write your CSR as if your imagined customer would write it! —Describe the problem that your customer is facing and his or her suggestions about how a software system could help.

Your CSR should be based on your project proposal, revised and improved as necessary. If you’re working on an existing project idea, then summarize and rephrase the description given therein.

You are welcome to borrow anything and everything from the past student projects posted there; just make sure that you describe explicitly how novel or different your extensions will be compared to the past projects.

# Glossary of Terms

List important terms and their definitions to ensure consistency and avoid ambiguity in the system specification. Use the language of the application domain and avoid uncommon terms or define these as well.  
It is helpful to illustrate the complex terms by providing images and graphics to help reader’s understanding ([find images on the web](http://www.google.com/imghp)).  
Another option is to provide web links where to find more complete definitions of your terms.

# System Requirements

Note: Instead of system requirements, you may wish to write User Stories (write one or the other, but not both).

## Enumerated Functional Requirements

Extract the requirements from the customer’s narrative and list them in a table, one row per requirement. The first column shows a unique label “REQ-x”. The second column shows an assigned Priority Weight of this requirement. The third column briefly describes the requirement.

## Enumerated Nonfunctional Requirements

List, prioritize, and describe the FURPS+ requirements. The non-functional requirements numbering should continue the functional requirements list.

## On-Screen Appearance Requirements

For projects that are heavy on graphics (such as a video game) the on-screen appearance makes up the majority of the requirements. Again list, prioritize, and describe the on-screen appearance requirements, but also include a graphic illustrating the requirement. You may find images on the Web or make hand-drawn sketches on paper, then scan them and insert as images into your report.

# Project Management and References

## Merging the Contributions from Individual Team Members

Compiling the final copy of the report from everyone’s work, ensuring consistency, uniform formatting and appearance.  
Describe what issues were encountered and how they were tackled.

## Project Coordination and Progress Report

What use cases have been implemented?   
What is already functional, what is currently being tackled?  
List and describe other relevant project management activities.

## History of Work

List the milestones and the dates they were accomplished. Preferably, you should use [Gantt charts](http://www.ganttchart.com/) for planning and scheduling your project.

## Breakdown of Responsibilities

* List the names of modules and classes that each team member is currently responsible for developing, coding, and testing
* Who will coordinate the integration?
* Who will perform and integration testing? (The assumption is that the unit testing will be done for each unit by the student who developed that unit.)

# Functional Requirements Specification

Elaborate only the use cases that will be implemented by the time of the final demo. For the use cases that will not be implemented for the final demo, provide a casual description for each and indicate that these could be considered for future work.   
System Sequence Diagrams should be updated to incorporate the use cases that will be completed for the final demo.  
This section must include the Traceability Matrix that shows how your use cases are related to your system requirements.

## Stakeholders

Identify anyone and everyone who has interest in this system (users, managers, sponsors, etc.). Stakeholders should be humans or human organizations.

## Actors and Goals

Identify the roles of people or devices that will directly interact with the system, their types (initiating vs. participating) and the goals of the initiating actors.

# Use Cases

## Casual Description

For **all** use cases that you can think of (based on your System Requirements), write a brief or casual text description. List explicitly the requirements that each use case responds to.

## Use Case Diagram

Draw the use case diagram with all the use cases. Indicate the relationships, such as <<include>> and <<extend>>.

## Traceability Matrix

Show how your system requirements map to your use cases. Calculate the priority weights of your use cases. The use cases with the highest priority should be elaborated and planned for the first demo.

## Fully-Dressed Description

Select a **few most important** use cases and provide detailed (“fully dressed”) description. The “most important” use cases are indicated by your traceability matrix.  
Your event flows must show step-by-step every action that the initiating actor (“user”) can take while running the given use case.

## System Sequence Diagrams

Draw the system sequence diagrams for the **few most important** use cases selected above.

# User Interface Specification

(Note: If your system prints some forms or generates periodic reports, this is also considered part of the user interface and the format of forms/reports must be specified in this section.) The user interface should be specified only for the use cases elaborated in the previous section (“fully dressed” use cases).

## Preliminary Design

For a given use case, show step-by-step how the user enters information and how the results appear on the screen.

Use screen mock-ups and describe exactly what fields the user enters and buttons the user presses. Describe navigational path that the user will follow. In case you are developing a graphics-heavy application, such as a video game, this is one of the most important sections of your report.

## User Effort Estimation

Select several typical usage scenarios and, as you walk through the flow of events, count and report the number of mouse clicks and/or keystrokes that are needed to accomplish the task. What fraction of these goes to user-interface navigation vs. clerical data entry?

# Domain Analysis

This section must include the Traceability Matrix that shows how your use cases map to your domain concepts. Include text description, not only a table with checkmarks.

## Domain Model

Show the process of deriving the domain model and then draw the diagram. Provide text description of:

1. Concept definitions
2. Association definitions
3. Attribute definitions
4. Traceability matrix — show how your use cases map to your domain concepts.

## System Operation Contracts

Should be provided only for the operations of the fully-dressed use cases.

## Mathematical Model

Do you use any mathematical models? E.g., you may use a statistical model for stock price prediction, or a geometric model for computing the trajectories for animate figures in a video game. If NO, skip to the next item; If YES, describe precisely your model.

# Interaction Diagrams

IMPORTANT: Your revised interaction diagrams must include some of the **Design Patterns** that were covered in the lectures after the Report #2 was submitted.   
Explain and justify the patterns that you use in your new design. State explicitly in what sense the use of the specific design pattern in the particular interaction diagrams improves the design.

Do [interaction diagrams](http://en.wikipedia.org/wiki/Unified_Modeling_Language#Interaction_diagrams) for the use cases you elaborated (“fully dressed”). You should do at least [sequence diagrams](http://en.wikipedia.org/wiki/Sequence_diagram), but you may do some other UML interaction diagrams, as well.

**Describe** what design principles you employ in the process of assigning responsibilities to objects. This can be done either as comment “bubbles” in the diagram, or in the caption of the diagram.

# Class Diagram and Interface Specification

This section must include the Traceability Matrix that shows how your classes are related to your domain concepts and a text description of the concepts-to-classes evolution.

## Class Diagram

Show all classes and their associations. Only indicate visibilities of attributes and operations; full details about the types and signatures should be provided in the next item.  
If you cannot fit the class diagram on one page, or it looks too cluttered, create one “overview” class diagram showing all classes and their relationships, but for each class show only a single compartment with the class name (leave out attributes and operations).  
Then on subsequent pages show partial class diagrams, with three compartments and all the attributes and operations of a class. Make sure to indicate in diagrams and describe in text how partial diagrams fit into the overall class diagram.

## Data Types and Operation Signatures

Independently of the class diagram, write down class specification in UML notation. For every class, specify data types of all attributes and operation signatures.  
**Define** the meaning of each class, operation, and attribute in plain language.

## Traceability Matrix

Show how your classes evolved from your domain concepts. Provide explanations for modified names or multiple classes that evolved from a single concept—providing only a matrix with checkmarks is not enough!  
Note that if your system has many classes, the matrix may become difficult to read. Alternatively, you may list, line-by-line, all your domain concepts and explain which classes were derived from each concept and why.

## [Object Constraint Language (OCL)](http://www.omg.org/spec/OCL/2.0/) Contracts

List important contracts (invariants, preconditions, postconditions) for classes and their operations.

# System Architecture and System Design

## Architectural Styles

Describe the [architectural styles](http://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.2F_Patterns) used in your design (Google search [software architectural styles example](http://www.google.com/search?hl=en&q=software+architectural+styles+example&aq=0&oq=software+Architectural+Styles))

## Identifying Subsystems

Draw and describe **UML package diagram** of subsystems in your system

## Mapping Subsystems to Hardware

Does you system need to run on multiple computers? For example, you may have a client (web browser) and a server (web server) subsystems, running on different machines.   
If NO, skip to the next item;   
If YES, what subsystems run on which machine?

## Persistent Data Storage

Does your system need to save data that need to outlive a single execution of the system?   
If NO, skip to the next item;   
If YES, identify the persistent objects and select the storage management strategy, e.g., flat files, relational database, etc.   
Attach the description of the file format and/or **database schema** (format of database tables, printed by the command description)

## Network Protocol

If your system runs on a single machine, this question probably does NOT apply; skip to the next item;   
Otherwise, which communication protocol do you use, e.g., plain Java sockets, Java RMI, Java JDBC, HTTP, etc. Explain why you made your specific choice.   
If you are using plain sockets, attach the description of the communication protocol (types of messages, message format, etc.).

## Global Control Flow

* Execution orderness: Is your system procedure-driven and executes in a “linear” fashion, where every user every time has to go through the same steps, or is it an event-driven system that waits in a loop for events, and every user can generate the actions in a different order?
* Time dependency: Is there any timers in your system?   
  Is your system of event-response type, with no concern for real time, or is it a real-time system? If it is real-time, is it periodic, and what are the time constraints for each period?
* Concurrency: Does your system use multiple threads?   
  If NO, skip to the next item;   
  If YES, identify the objects that have separate threads of control and describe any synchronization between the threads?

## Hardware Requirements

What system does resources your system depend upon? Examples are, screen display, disk storage, communication network, or you may be accessing some special sensor/instrument.   
Describe exact requirements for these resources in order for your system to run. For example, you need color display, with minimum resolution of 640 × 480 pixels; minimum of 2 Gbytes hard disk space; minimum network bandwidth 56 Kbps.

# Algorithms and Data Structures

## Algorithms

Describe the algorithms that implement mathematical models from your Report #1. Does your system use any other complex algorithms? For example, when computing a motion trajectory for an animate figure in a game, you may use some numerical or computer-graphics algorithms. Or, when assessing stock market movements, you may be using statistical algorithms.   
If NO, skip to the next item;   
If YES, describe your algorithms. For example, for the animate figure example above, will the path coordinates be precomputed and stored in a look-up table or will they be computed using a spline interpolation algorithm.   
It is a good idea to use activity diagrams to describe the algorithm design.

## Data Structures

Does your system use any complex data structures, such as arrays, linked lists, hash tables, or trees?   
If NO, skip to the next item;   
If YES, what criteria you used in deciding what data structure to use, e.g., performance vs. flexibility?

# User Interface Design and Implementation

Describe whether and how you modified and implemented the initial screen mock-ups developed. Comment only on significant changes in your user interface, those that reduce (or increase) the user effort. Changes of colors or styles are less important and should be omitted from your report.

“Ease-of-use” is generally considered a key characteristic of user interface. “Ease-of-use” should not be confused with a flashy interface, with lots of colors, picture, graphics, etc. On the contrary, you should avoid flashy user interfaces. “Ease-of-use” means that interface is intuitive, easy to understand and operate, without having to ask many questions or read voluminous documentation. A minimal user interface that is well organized should be sufficient.

# Design of Tests

* Note that for this report you are just designing your tests; you will program and run those tests as part of work for your first demo
* List and describe the test cases that will be programmed and used for unit testing of your software.
* Discuss the test coverage of your tests.
* Describe your Integration Testing strategy and plans on how you will conduct it.
* Describe also your plans for testing any algorithms, non-functional requirements, or user interface requirements.

# References

The list of references should contain exact references and URLs of any material that is used in the project and does not come from the textbook.

# Reflective Essays

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**

## Team Member: <Name>

A minimum 1-page must be included by **each student in the team.**