# Deliverable 2


## Team SAAB

**Bryce Bowser**

**Alex Paulson**

**Alex Upton**

Section 1 - Requirements Specification

# Software Requirements Specification

## For

# Tournament Manager

**Version 1.0 approved**

**Prepared by Alex Paulson, Alex Upton & Bryce Bowser**

**S.A.A.B.**

**10/11/2014**

# Table of Contents



# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| **Alex Paulson** | 10/11/14 | Initial Creation | 1.0 |
| | | | |

# 1. Introduction

## 1.1 Purpose

*Currently in development towards version 1.0, Tournament Manager is a program that was born in the problem solving school of thought. We had to make a program, so why not make one that will see use beyond the scope of the class?*

*Tournament Manager uses SQL databases, an ASP.NET web front and an intuitive GUI to give a tournament director complete control over a bracket-style, single or multi-room tournament. It is flexible to bracket size/style, manages teams' position in the bracket (and associated room, if multi-room) and allows for online team application.*

## 1.2 Document Conventions

*No special conventions used. Will update if this changes.*

## 1.3 Intended Audience and Reading Suggestions

*The design of this requirements document is such that it would be beneficial to potential users of Tournament Manager, but is necessary for the professor of the class. Anybody interested in the program for development or use purposes would benefit from scanning over the requirements document.*

*The layout of this requirements document is done as such to be easy to read, in-depth and flexible to change. No doubt, throughout the process of development this document will need to change, be it addition, subtraction or overhaul of information.*

## 1.4 Project Scope

*The idea of Tournament Manager was born out of a need of our friend who had problems hosting League of Legends tournaments. The tournaments were each held in multiple rooms and were bracket-style. The problem became that moving teams into other rooms to face their next opponent was a logistical nightmare when you factored in that time was a factor. Often times the tournament ran way too long and/or were forced to run over a 2nd day.*

*Thus, Tournament Manager was born. It has a flexible bracket design that will fit to many sizes and styles of tournaments. It keeps track of the positions of each team on the bracket and what room they're in (if the tournament is multi-room). Optionally, it uses an ASP.NET web front in order to allow teams to apply to the tournament ahead of time, reducing the managing load on the tournament director.*

# 2. Overall Description

## 2.1 Product Perspective

*Tournament Manager is a self-contained and complete program. It is not associated with nor part of any other project.*

## 2.2 Product Features

*ASP.NET Web Front – A web front provided for the optional feature of letting teams apply online, and therefore submit their information into the database, ahead of the tournament, reducing the load on the tournament director.*

*Bracket Manager – A flexible component that keeps track of teams on a bracket of variable size and style (single, double elimination, round robin, etc.). Simplicity is a goal, but it aims to be a powerful tool.*

*Room Manager – If necessary, the tournament director may use the room manager feature, which ties a room to each team and helps efficiently sort the teams graphically so that teams may get to their next matchup quicker.*

## 2.3 User Classes and Characteristics

*The Tournament Director is the only level of user class we're gearing the program towards. If he choses to let somebody else run the program, they will assume the same abilities and permissions as the director. It is a single set of permissions required to run the program. No more are necessary, and any less would inhibit the program operation.*

## 2.4 Operating Environment

*Tournament Manager will be strictly a Windows program developed on the Visual Studio C# platform. It will run most likely on a laptop, though the hardware involved won't matter as long as the device is running Windows. The program is lightweight and, other than an internet connection for the optional web front, has very, very minimal requirements. Updated .NET framework may have to be downloaded in order to run, though most systems already have that in place/taken care of through updating.*

## 2.5 Design and Implementation Constraints

*Tournament Manager is both written and designed in such a way that it will run on virtually every Windows device. It is easy on hardware and has almost no special requirements. The*

*tournament director will be able to run the program with little necessary setup (save an internet connection if they use the optional web front).*

## 2.6 User Documentation

*Presence/lack of user documentation, along with specific style and details is TBD.*

## 2.7 Assumptions and Dependencies

*Tournament Manager was designed under the premise that the tournament director need only to have a good understanding under how his tournament is designed, so that he can effectively set up Tournament Manager for their unique tournament. The design is such that beyond that, the director need not know much at all about how Tournament Manager operates, and therefore need not make assumptions. This level of simplicity is in place to help ensure smooth operation.*

# 3. System Features

## 3.1 Bracket Manager

### 3.1.1 Description and Priority

*See 2.2*
*Core feature necessary to operation. High priority.*

### 3.1.2 Stimulus/Response Sequences

*Initially, once a bracket size and style are selected, teams are placed on spots on the bracket either at random or by the director. As teams beat their opposition, the director(s) job is to update the bracket by selecting which team wins, which pushes them along the bracket.*

### 3.1.3 Functional Requirements

*REQ-1: Windows Operating System*
*REQ-2: Updated .NET Framework*
*REQ-3: Be flexible to bracket size and design*
*REQ-4: Allow simplistic moving of teams along bracket*
See 4.3

## 3.2 Room Manager

### 3.2.1 Description and Priority

*See 2.2*
*Core feature necessary to operation. High priority.*

### 3.2.2 Stimulus/Response Sequences

*Responding in kind to the same selection process that the director makes to move a team along the bracket, the Room Manager feature keeps track of what teams are in which rooms and helps show where they have to get to next.*

#### 3.2.3 Functional Requirements

*REQ-1: Windows Operating System*
*REQ-2: Updated .NET Framework*
*REQ-3: Move teams based upon bracket location/seed*
*REQ-4: Efficiently move teams to speed up tournament*
*See 4.3*

## 3.3 ASP.NET Web Front

#### 3.3.1 Description and Priority

*See 2.2*
*Can function without web front IF necessary. High priority, but can change.*

#### 3.3.2 Stimulus/Response Sequences

*Teams enter their information onto the web front ahead of the tournament. The information is stored into an SQL database and is then available to the director for a pool of teams at the start of the tournament.*

#### 3.3.3 Functional Requirements

*REQ-1: Windows Operating System*
*REQ-2: Updated .NET Framework*
*REQ-3: Internet Connection*
*REQ-4: Be secure in payment transactions*
*REQ-5: Allow only proper input format(s)*
*REQ-6: Transfer data into appropriate fields in DB*
*See 4.3*

# 4. External Interface Requirements

## 4.1 User Interfaces

*Each of the 2 separate managers, Bracket Manager and Room Manager, have a different interface. Both are intuitively designed to be able to be displayed on a TV or through a projector and be easily read by all attendees at the tournament. Simplicity is the driving design factor that helps produce this effect.*

## 4.2 Hardware Interfaces

*Tournament Manager is designed from the ground up to be lightweight and not require anything of the hardware except that it be an x86/64 machine running Windows.*

## 4.3 Software Interfaces

*The main Tournament Manager program only needs to be able to communicate with a Windows operating system (which is the only environment it will run on). The functional user requirements on the GUI are as follows:*

- *Allow the user to sign up and submit/pay for a team (through web front GUI, if used)*
- *Allow the tournament director to move teams along the bracket by selecting a winner*
- *To notify the tournament director of room changes necessary to continue matches (may also be cast upon a projector)*
- *Allow the tournament manager to create a bracket that is correctly designed and sized for the tournament*
- *Move teams on the room manager in an efficient order in regards to place on the bracket*

*Non-functional requirements:*

- *Be simplistic in design*
- *Be efficient and effective*
- *Allow for flexibility*
- *Have project be well documented*

## 4.4 Communications Interfaces

*Specifics TBD, but the ASP.NET web front is going to be the online part requiring communications, as the rest of the program is designed to be run without an internet connection.*

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

*The program is designed to be lightweight while the problem it's fixing (lack of managing tools for tournaments) doesn't lend itself to need heavy computational work. Virtually any machine should be capable of running the program equally well.*

## 5.2 Safety Requirements

*Does not have any safety hazards. Used to speed up and automate decision making for tournaments but isn't a requirement and cannot worsen/slow the tournament experience.*

## 5.3 Security Requirements

*Only security requirements are rested with the tournament director. They need to be sure that they only allow access to the machine running to themselves and/or trusted individuals. ASP.NET web front will require some security. Using a third party (such as PayPal) to handle transactions will help negate some possible security flaws. Fine details TBD.*

## 5.4 Software Quality Attributes

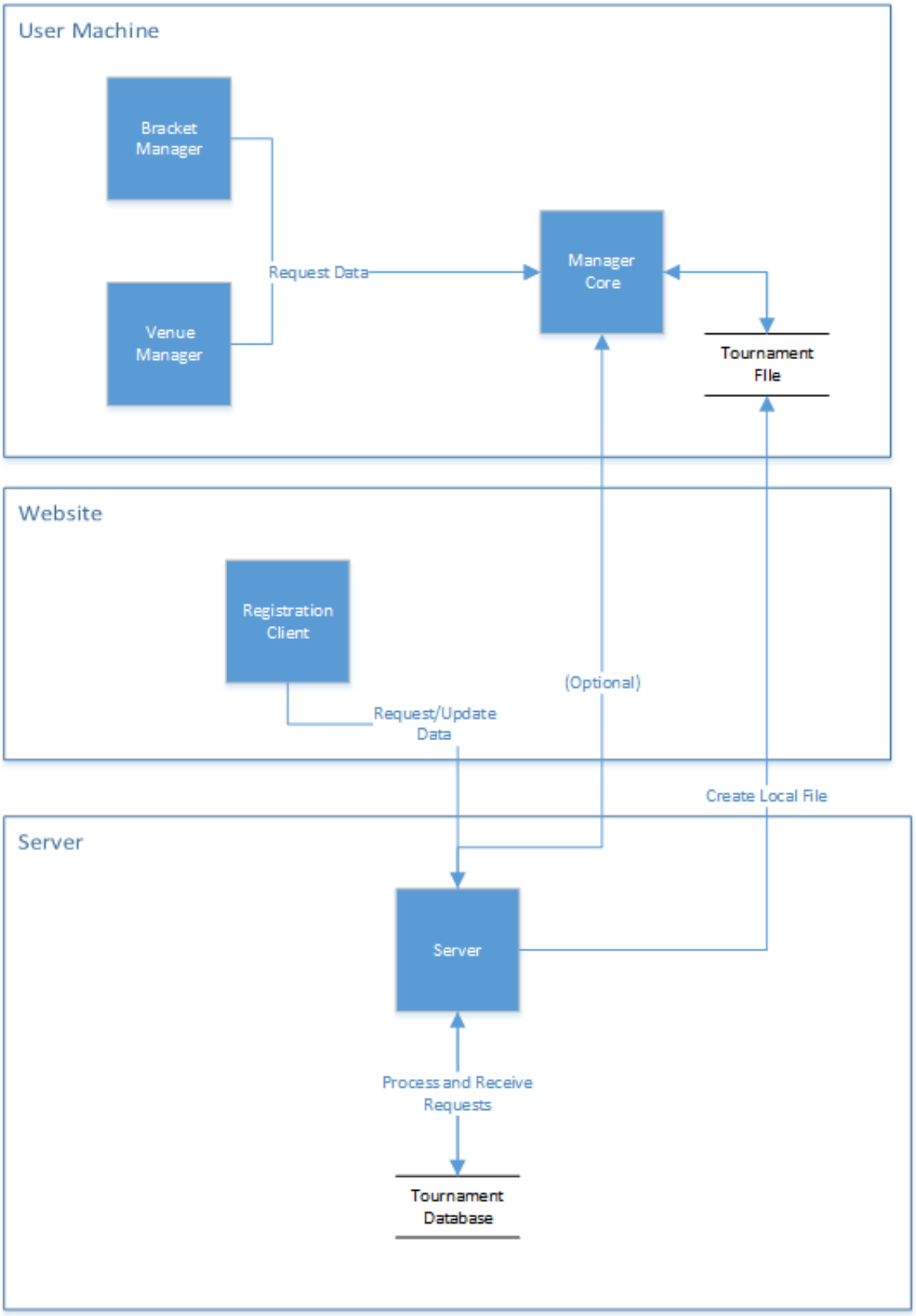
*Tournament Manager is designed to be quick and simple. It (aside from the web front, which is completed before the tournament anyways) is completely independent from other systems and is designed to be portable and able to be run on any device running Windows.*

## Section 2 - System Architecture

Tournament Manager features a client-server model architecture with both a web front application and a standalone local application communicating with the same team database. In the Registration client, administrators can customize tournaments in several powerful ways, including setting team size, game size, and registration fee amount. These functions correspond to the applicable database fields. The registration client also allows tournament participants to view and register for a specific tournament.

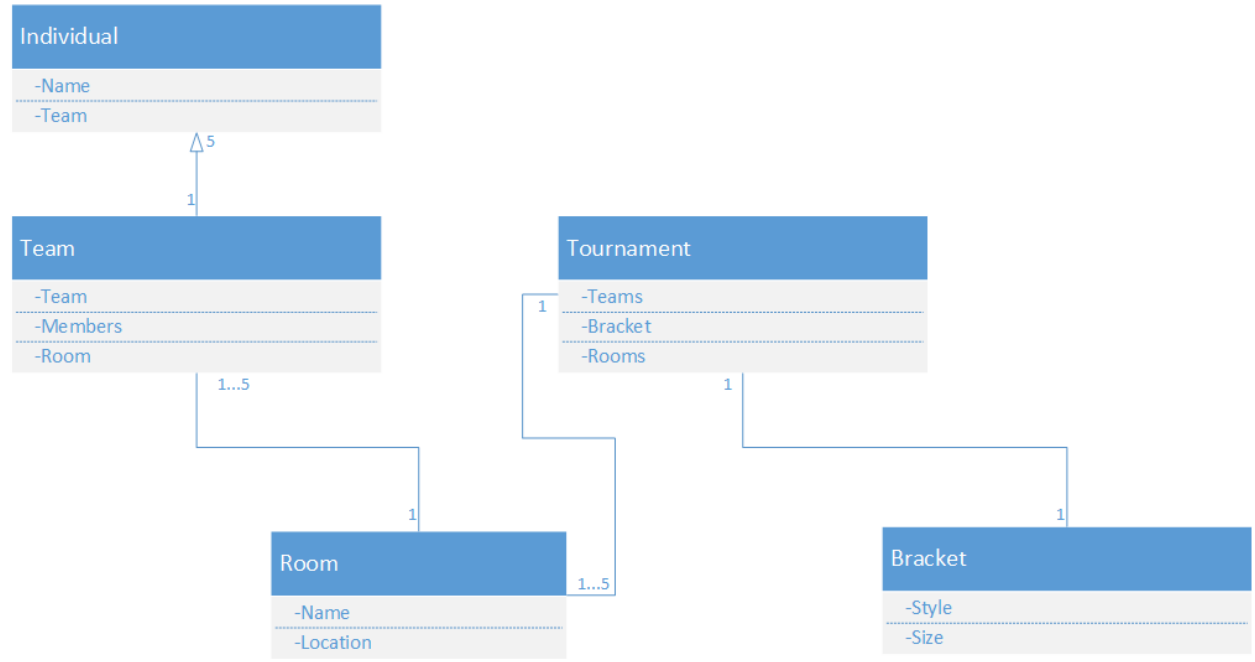
The Bracket Manager and Venue/Room Manager are separate functions that extend the functionality of the Registration Client by allowing participants to check in at the tournament and receive venue assignments. The standalone client also includes all the features of the Registration client. Bracket and Venue Manager require an internet connection to access the online database, or a local tournament file containing a hard-copy backup of the tournament data downloaded by the Registration Client.

**Fig. 2-1 System Architecture Overview**

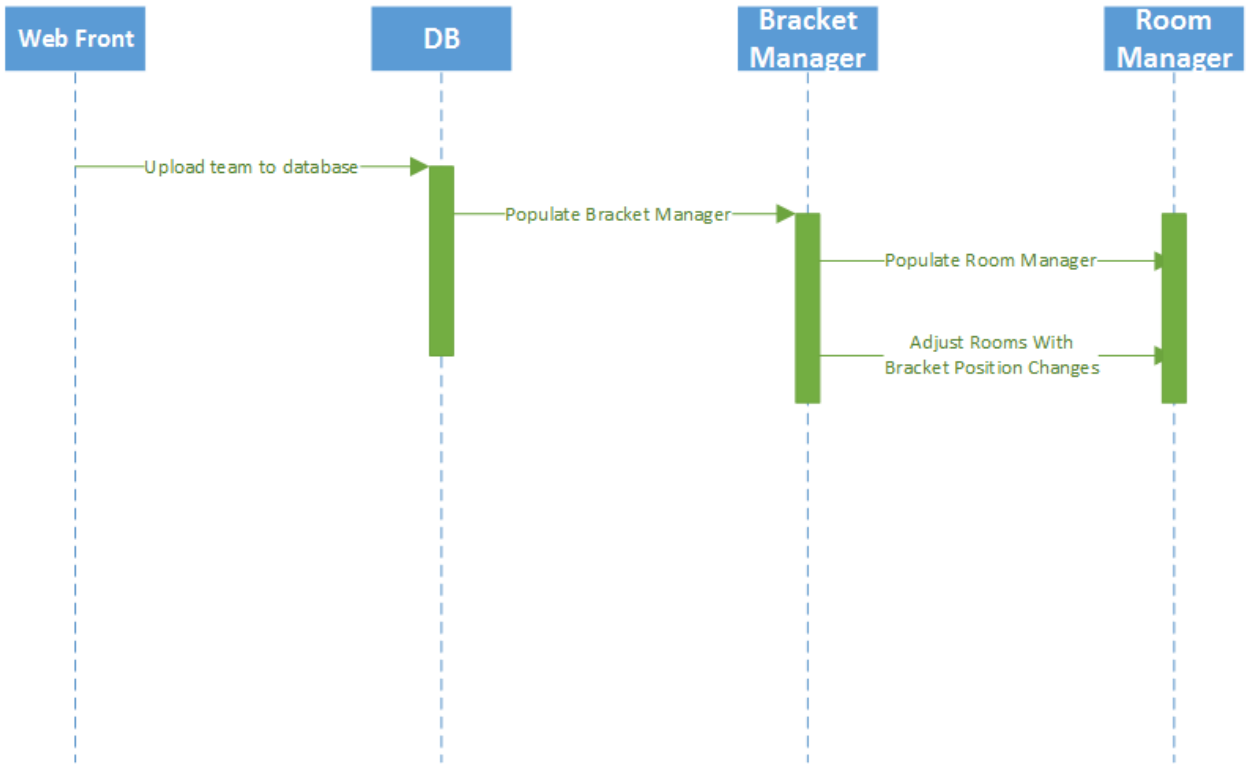


# Section 3 - UML Diagrams

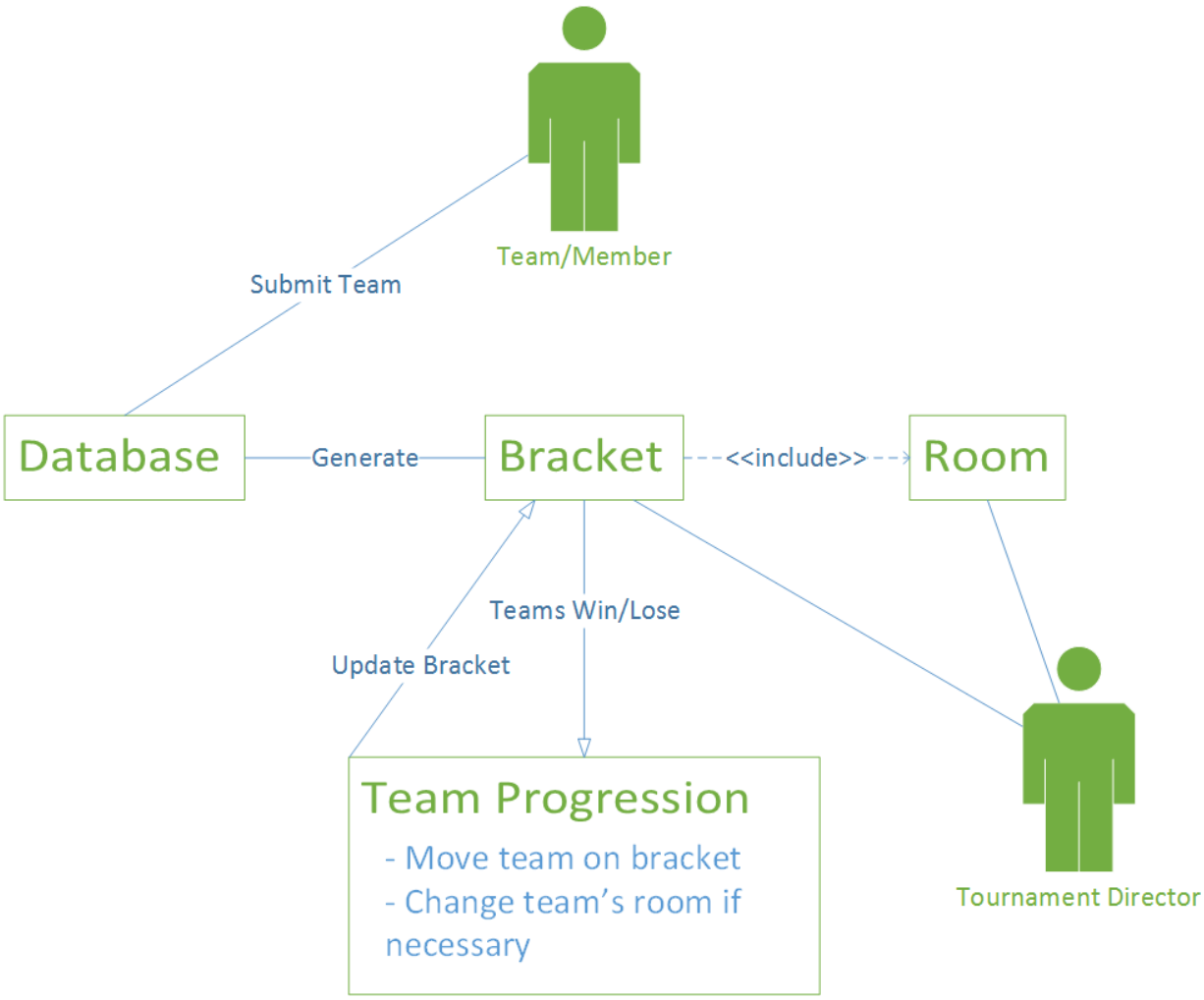
### Class Diagram




### Sequence Diagram



**Use Case Diagram**



## Section 4 - Test Plan

To ensure proper functionality tests will be performed in three separate phases.

1. Unit Level
2. Subsystem Level
3. System Level

Each of these three levels represents a larger portion of the completed code. This sequence is structured in such a way that simpler functions are verified and validated first before introducing a larger and more complex combination of functions, thus reducing the need for fixing simple features in the later more costly stages of development.

**Unit Level:** This is the most fundamental level of testing. Functions and methods will be tested for accuracy and completeness. Simple functions such as creating objects and collecting inputs will be tested first for correctness and validity. This testing will be performed by the individual writing the code for that segment.

**Subsystem Level:** Individual modules of the system will be tested in the subsystem phase. Each of the three components will be tested separately to ensure optimal performance before moving on to system level integration testing.

For the Registration Client, unit level testing needs to verify and validate the following features

- Create Tournament
    - Collect data from fields
    - Insert the data correctly into the database
- Register Team
    - Evaluate team for conformity to tournament properties
    - Insert the data correctly into the database
- Delete Tournament
- Edit Tournament

For the Manager Core and related programs, the following features will be tested

- Read from sample database
    - Create correct objects from data
- Populate and update tournament bracket
    - Advance winning teams
    - Use seed data if available to generate bracket positions
    - Allow manual editing of brackets by administrator
- Assign venues efficiently
    - Assign teams to same venue in Classic Mode, different venues in ESports Mode
    - Display graphical representation of teams and venues
    - Allow manual editing of venues by administrator

**System Level:** Once the units and subsystems have been verified and validated, the final system-level testing will evaluate the shared functions of the entire system. This will include:

- Communication between the Registration Client and the server and database.
    - Concurrency, safety and load testing
- Communication between the Manager Core and the server
- Ability to create local files from the Registration Client to be used by the manager
- Reading and modifying tournament files in the Bracket and Venue Manager
- Client evaluation - Conforms to client expectations

## Section 5 - Risk Management (Update)

**Identify:**

1. **Time Management:** Time management is going to be key in any project and presents the risk of letting the project getting behind schedule if time is mismanaged.

With writing the project in C#, it allows to cut down on the time needed by being able to create a database and a GUI all in one.

2. **Feature Planning:** One large part about any project is planning, obviously. However, one part worth noting in this section is the planning of features. We, as a group, and especially the coders, need to agree on a slate of features in both how they're made and what they do. This allows us to get to work on the project/features without having to argue about what parts we do/don't want to include.

Already, thanks to a fairly common understanding of what is going into the project, plus a simplistic approach to the program, we have gotten a great jump on the project and have worked together with what needs to be done.

3. **Identification of Strengths:** One key part of the project, that can lead to problems if not accurately completed, is the self-assessment of one's strengths and weaknesses, and the distribution of duties based upon those assessments.

We have performed our strengths to perfection, and everybody is happy doing what they are assigned.

4. **Project Management:** An imperative part of software engineering is to be clear and concise about what parts of the projects are whose responsibilities. Risk is presented if people try to do too much or parts don't get done/get done twice due to mismanagement of duties.

Everybody in the group has had a clear understanding of what they have needed to accomplish and have done well thus-far.

5. **Collaboration:** An idea stemming from project management, it is necessary as a team to collaborate effectively and efficiently on our project in order to maximize the efficiency of time spent individually. If we are able to work on goals that mesh together, rather than working on parts at random, we can cut down on time spent debugging and reworking code.
We have worked as a group well so far. It hasn't cropped up as an issue.

## Section 6 - Project Plan

(Red = date passed)
9/7/14 – Get together, work on requirements and finalize Deliverable I.
9/14/14 – Prepare UML Diagram and design.
9/18/14 – Craft game plan for project/order of importance.
9/24/14 – Miscellaneous work, SQL database start.
9/30/14 – Work on Deliverable II and see where everyone is at with their duties.
10/11/14 – Work on Deliverable II again.
10/13/14 – Finalize Deliverable II.
10/23/14 – Work on the Integration with the database.
11/4/14 – Wrap up coding and troubleshoot the program.
11/16/14 – Start preparing presentation.
11/23/14 – Final testing of program.
11/25/14 – Work on Deliverable III.
12/1/14 – Finalize Deliverable III.
12/4/14 – Deployment of code.
12/6-7/14 – Prepare and practice the project presentation.

## Section 7 - Meeting Minutes

## Meeting Schedule

9/7/14 – Get together, work on requirements and finalize Deliverable I. **ATTENDED. (45 MINS)**
9/14/14 – Prepare UML Diagram and design. **ATTENDED. (90 MINS)**
9/18/14 – Craft game plan for project/order of importance. **ATTENDED. (60 MINS)**
9/24/14 – Miscellaneous work, SQL database start. **ATTENDED. (90 MINS)**
9/30/14 – Work on Deliverable II and see where everyone is at with their duties. **ATTENDED. (30 MINS)**
10/11/14 – Work on Deliverable II again. **ADDED AND ATTENDED. (45 MINS)**
10/13/14 – Finalize Deliverable II. **ATTENDED. (30 MINS)**

## Section 8 - Progress Report

Insofar Tournament Manager has been largely in the planning phase. Coding has just started. Everything is on time according to the schedule. We plan on working on/finishing the mainframe of the project (the Bracket Manager and Room Manager) before implementing the web front. That way, if time ends up being a constraint, we will still have a fully functioning and working product, simply without the optional web front.