Modelario

A Software Development Team Project

Prepared by
Group Modelario
Justine Saylor, Michael Chase Townsend, Brayden Reece, and Adrian Ray

April 23rd, 2023

Table of Contents

List of Figures

List of Tables

I Phase 1 – Project Concept and Plan

1 Problem / Project Description

To make a web browser based multiplayer game where one player is given a prompt and that player then draws the prompt on a digital canvas while the remaining players proceed to guess what the prompt was. The player who guesses the prompt correctly is then awarded points that are then stored digitally.

2 Software Development Life Cycle (SDLC)

For our project we chose the SCRUM method for our SDLC, mainly due to its agile method tendencies along with its promoting of transparency and communication among team members, which leads to better collaboration and more effective problem-solving.

3 Project Scope

This web-based project would be limited to an online platform that users can access through their web browsers. There is not a limit on the number of players and the drawing is limited to a 2-D drawing using the shapes and tools provided based on the given prompt. The game is limited to a browser that supports Java Script and WebGL and is limited to a desktop application and cannot be used on mobile devices. Deliverables would include a functional web based application that meets the requirements that are later defined, technical and or user documentation, and a plan to ensure the application works as expected.

4 Project Team Organization

Team members are Justine Saylor, Michael Chase Townsend, Brayden Reece, and Adrian Ray. Justine's main role will be the librarian but will also help with the requirement analyst's role and configuration management. Chase's main role will be the project manager and programmer but will also help with training. Brayden will be the programmer, but he will also help with training. Adrian's main role will be website designing and testing. We will all collaborate and contribute to designing, testing, and maintaining the software. We have a loosely structured team with an egoless approach where every team member is responsible for their own deliverable parts, and we are all equally responsible. For resource allocation, we will be reusing parts of software created by team members for part of our web application.

5 Technical Description of the System

The technical aspects of this project can be broken down into two categories, those being the website and the game itself. There is going to be only one version of the game which is the initial prototype.

5a Technical Description for Game

Software:

- HTML/CSS for displayed UI.
- JavaScript for the logic of the game.
- WebGL for the canvas and drawing aspect.
- The game will also be implemented using an OOP style.

Hardware:

- Can be run on any modern PC.

Restrictions:

- WebGL is limited to very select primitive shape types.

5b Technical Description for Website

Software:

- HTML/CSS for website implementation.
- PHP files to establish client to server paradigm.
- JavaScript for other miscellaneous tasks.

Hardware:

- The server-side aspect will be hosted on one of the developers' devices.

Restrictions:

- No real implementation for security.
- The application uptime is dependent on the host device.

6 Project Standards and Procedures

The main important standard that should be followed is that of readable code. Since this project is for a group or simulated company, code readability, such as not having spaghetti code, as well as consistent code formatting for all programming, is extremely important to output an effective product. Some examples of specific formatting techniques that will be used for this project are how new functions/curly braces are formatted. For this project, every beginning curly brace will be placed on a new line, and the function code will be indented by one tab. Another example is one line if statements will not include curly braces. These formatting systems may seem irrelevant, but they actually go a long way in having consistent and readable code understood by all developers. This also, in turn, helps with the functional testing method, as it makes looking for bugs and code functionality easier since everything will be formatted in the same manner.

7 Documentation Plan

The documents that will be produced include the project concept and the project plan, which are part of phase one. Phase one final documents will be produced on March 16th. Other documents that will be produced are the list of requirements in phase two, the

design in phase three, and the final project with testing results in phase four. All documents will be produced with contributions from each team member, but Justine will produce the final formatted document for each phase. All documents will be produced by their due dates depending on each phase. Another document that will be produced is the master document for everyone to share their work, and meeting agendas for each meeting. The meeting agendas are produced by Chase before each meeting, and everyone collaborates on the master document, which will be completed at the end of the project.

8 Data Management Plan

Data in this project will be stored non-persistently, and the important values being stored are as follows: player names, player scores per round, and player scores overall. Though this data may be stored persistently in future iterations of the game.

9 Resource Management Plan

We plan to minimize the physical resources used by our application on the computer by profiling our code and finding spots that take up large amounts of resources, then optimizing them by changing or improving the current algorithm to be less resource intensive, given the current workload. Some examples of these efforts will include stripping unused or unnecessary variables from the code, utilizing the best algorithm for the current workload, and letting performance speak for itself.

10 Security Plan

We will use the built-in security of Google, Discord, and GitHub since these are our primary tools for our activities. As a team, we lack the time or knowledge to expand our security abilities further. Our approach to security will be to build what we can where we can for validation and safety but to trust the current systems already in place. We are keeping score only for each individual session so we can avoid having a login with credentials as a possible security issue.

11 Risk Management Plan

One possible risk we have identified is our web application game using too many resources, causing the website to crash. This could be due to the different tools and features of the painting canvas. If this risk were to occur, it would have the second highest priority. A way to reduce this risk is to avoid the risk by changing the requirements for performance or functionality. Another possible risk we have identified is a possible data breach of user information if we don’t secure our website. If this risk were to occur, it would have the highest priority as we want to protect our user’s data. A way to reduce this risk is to assume the risk and accept and control it. As we develop the project, we will continue to monitor for risks and reassess as we progress.

12 Maintenance Plan

We have a small team and two programmers, so our general plan will be to keep the code compact enough to look over and roughly understand. To this end, if a code file reaches more than 500 lines of code, it will be split into two different files along lines that make sense for the application. We do not have training materials or plans to generate any since we have the requisite knowledge in the relevant areas and will not accept new members now.

II Phase 2 – Software Requirements Specification (SRS) Document

1 Introduction to the Document

1a Purpose of the Product

To make a web browser-based multiplayer game where one player is given a prompt, and that player then draws the prompt on a digital canvas while the remaining players proceed to guess what the prompt is. The player who guesses the prompt correctly is then awarded points that are then stored digitally.

1b Scope of the Product

This web-based project would be limited to an online platform that users can access through their web browsers. There is not a limit on the number of players and the drawing is limited to a 2-D drawing using the shapes and tools provided based on the given prompt. The game is limited to a browser that supports Java Script and WebGL and is limited to a desktop application and cannot be used on mobile devices. Deliverables would include a functional web based application that meets the requirements that are later defined, technical and or user documentation, and a plan to ensure the application works as expected.

1c Acronyms, Abbreviations, Definitions

Canvas - A drawable area that users can modify by adding shapes with colors.

Shape - A regular polygon, circle, triangle, or rectangle that can be specified by a number of clicks on the canvas.

1d References

Other related systems include Pictionary and Skribbl.io.

1e Outline of the Rest of the SRS

II. General Description of Product

- A. Context of Product
- B. Product Functions
- C. User Characteristics
- D. Constraints
- E. Assumption and Dependencies

III. Specific Requirements

A. External interface requirements
    1. User Interfaces
    2. Hardware Interfaces
    3. Software Interfaces
    4. Communications Interfaces
B. Functional Requirements
C. Performance Requirements
D. Design Constraints
E. Quality Requirement
F. Other Requirements

IV. Appendices

2 General Description of Product

2a Context of Product

This software allows users to create drawings based on prompts while competing against other players. Our system allows users to play with multiple users while only using one device which is a different approach compared to systems that currently exist.

2b Product Functions

- Drawing
- Scoring
- Scorekeeping
- Win/Lose
- Timer

2c User Characteristics

- Inexperienced user that has previously limited experience with computers.

2d Constraints

- Mouse and Keyboard.
- Must comply with COPPA.

2e Assumption and Dependencies

- Assume that the user has a constant internet connection.
- Assuming the user has the latest browser update for respective browser.
- Computers may crash due to resource overload.

3 Specific Requirements

3a External Interface Requirements

User Interfaces

- HTML / CSS.

Hardware Interfaces

- Any Modern PC with a standard Keyboard and mouse.
- The server-side aspect will be hosted on one of the developers' devices.

Software Interfaces

- HTML/CSS for displayed UI.
- JavaScript for the logic of the game.
- WebGL for the canvas and drawing aspect.
- HTML/CSS for website implementation.
- PHP files to establish client to server paradigm.
- JavaScript for other miscellaneous tasks.

Communications Interfaces

- WebGL.
- JS standard Library.
- HTML / CSS.
- User Interface.

3b Functional Requirements

- Must be able to select and draw one of the several following shapes:
    - Circle
    - Rectangle
    - Line
    - Triangle
- All shapes must be able to be any user selected color using the pinwheel.
- All shapes must either be filled in or have an outline.
- The first click must generate a ghost outline of the final shape.
- The nth click required to form the shape must finalize the shape as outlined by the ghost outline.

3c Performance Requirements

- Must not bog down the computer or make the computer hard to operate.
- Must not force the computer to operate with less than 30 frames per second on the hardware listed below:
    - Ram: 32 GB
    - Processor: Intel Core i7-8750H
    - Graphics Processor: 5 GB VRAM Nvidia GTX 1650
    - OS: Windows 10 Home Editions
    - Storage: 1 TB SSD

3d Design Constraints

- No effective way to implement security without an API.
- The application uptime is dependent on the host device.
- The game will be implemented using OOP.
- WebGL is limited to very select primitive shape types.

3e Quality Requirement

All shapes behind other shapes should be culled and the system should maintain pace with the user as they are drawing on the canvas.

4 UML Use Case Diagram

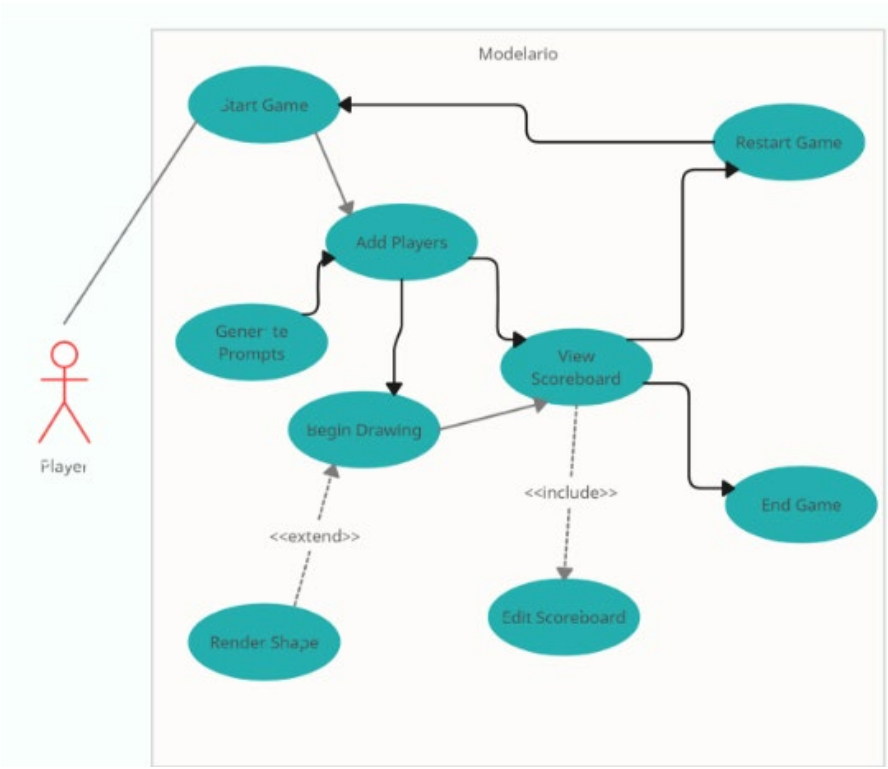

Figure 1 – High-level UML Use Case Diagram that specifies user views of essential system behavior.

III Phase 3 – Software Design Specification (SDS) Document

1 Frontspiece

1a Date of Issue and Status

April 11th, 2023

Active Status

1b Issuing Organization

Modelario Group

1c Authorship

Justine Saylor

1d Change History

Initial Draft 4-10-23

Second Draft 4-11-23

Final Draft 4-25-23

2 Introduction

2a Purpose

To make a web browser-based multiplayer game where one player is given a prompt, and that player then draws the prompt on a digital canvas while the remaining players proceed to guess what the prompt is. The player who guesses the prompt correctly is then awarded points that are then stored digitally.

2b Scope

This web-based project would be limited to an online platform that users can access through their web browsers. There is not a limit on the number of players and the drawing is limited to a 2-D drawing using the shapes and tools provided based on the given prompt. The game is limited to a browser that supports Java Script and WebGL and is limited to a desktop application and cannot be used on mobile devices. Deliverables would include a functional web based application that meets the requirements that are later defined, technical and or user documentation, and a plan to ensure the application works as expected.

2c Context

This software allows users to create drawings based on prompts while competing against other players. Our system allows users to play with multiple users while only using one device which is a different approach compared to systems that currently exist.

2d Summary

A web-based application game that gives a user a prompt in which the player must create a drawing using the given tools and shapes. The goal of the game is to get the most points by correctly guessing other players' drawings.

3 References

Other related systems include Pictionary and Skribbl.io.

4 Glossary

Canvas - A drawable area that users can modify by adding shapes with colors.

Shape - A regular polygon, circle, triangle, or rectangle that can be specified by a number of clicks on the canvas.

5 Body

5a Identified Stakeholders and Design Concerns

The stakeholders include the user, customer, and developer. In our case the user would be people that use our web application to play the game. The customer would be our professor since he prompted the system to be developed. The developers would be the members of our group (Justine Saylor, Brayden Reece, Adrian Ray, Michael Chase Townsend). One of our design concerns is the performance of our application, possibly bogging down the computer or making the computer hard to operate. Another one of our design concerns is the quality of the shapes in our application. In our application all shapes behind other shapes should be culled and the system should maintain pace with the user as they are drawing on the canvas.

5b Design Viewpoint 1

High-Level Architectural Design Viewpoint

5c Design View 1

Below are shown the major components of our system through both a dependencies view and an execution view.

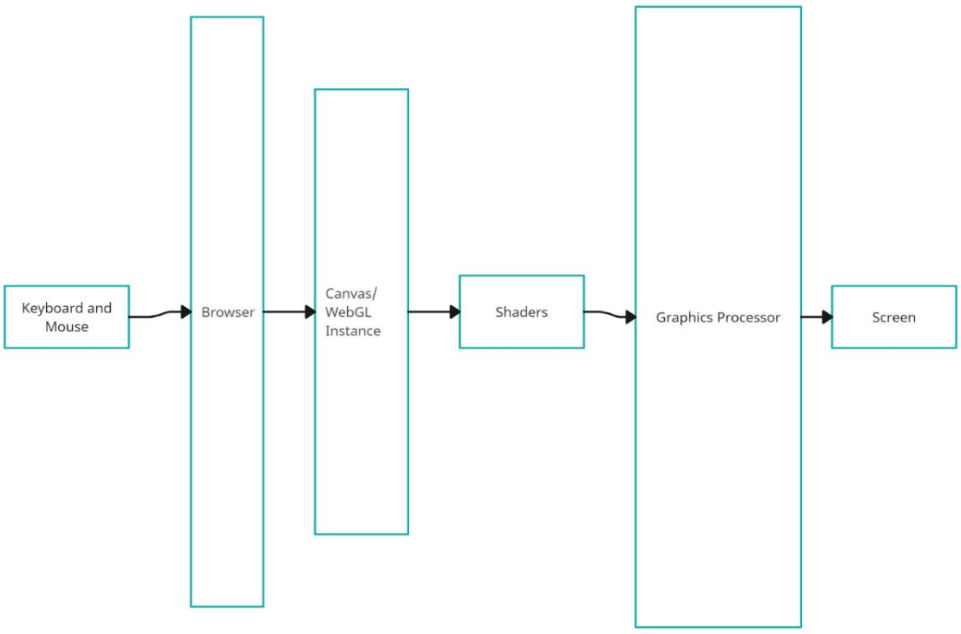

Figure 2 – Execution View of our System

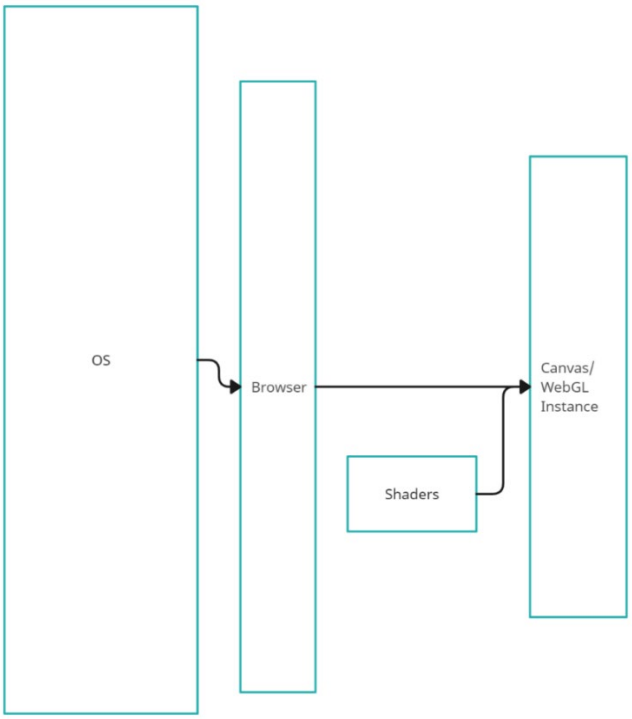

Figure 3 – Dependencies View of our System

5c1 Description of Major Components In Diagrams

Execution View Components

- Browser: Host for Canvas/WebGL instance.
- WebGL Instance: In browser graphics rendering API.
- Shaders: Name for programs that run on graphics processors.
- Graphics Processor: Compiles shaders into output able images for screen.

Dependencies View Components

- OS: Operating system of current user.
- Browser: Depends on OS to host Canvas/WebGL instance.
- Shaders: Name for programs run on graphics processor that depend on canvas/WebGL instance.
- WebGL instance: In browser graphics rendering API.

5c2 Architectural Styles

The architectural styles that apply to our project are pipes-and-filter, client-server, and publish-subscribe. The client-server style is used for our service allowing users to access the canvas and drawing tools using the web browser. Pipes-and-filters is used in the pipeline for the graphics specifically though the OpenGL Shading Language (GLSL) and makes system evolution simple. Publish-subscribe is used to read events that occur after users change the scoreboard.

5c3 System Interfaces

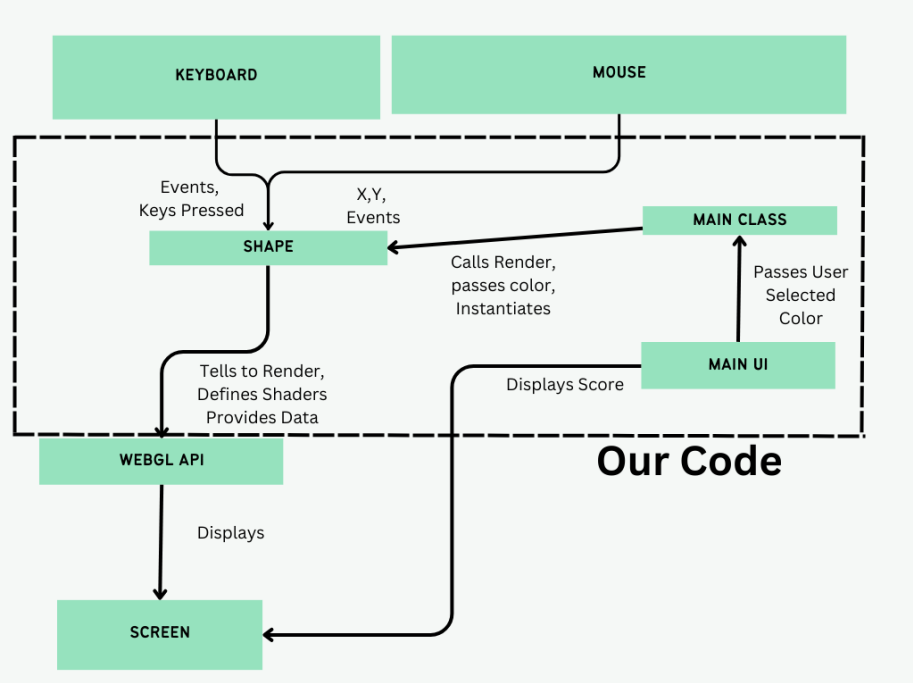

Figure 4 – Diagram of system interfaces. Shape – Shapes family of classes. Main Class – Main execution class. Main UI – Scoreboard, shape type picker, and color picker.

Each of the interfaces are described in section 5.5.3 and how the major components interact with each other are demonstrated in the diagram.

5c4 Design Issues

The design issues relevant to our project include reliability, reusability, maintainability, performance, portability, and security. Reliability is an issue when there are problems with the user's internet connection, but we cannot change our design in a way to prevent this issue. Reusability is an issue as there is no persistent storage within our web application. For this issue we designed our system where score is based on an individual game session and there is no overall score. Maintainability is an issue due to needing funding to maintain a host site for our web application. For this issue we designed our system to where we can maintain the web application ourselves rather than going through a host site. Performance is an issue if players draw too many shapes, taking up too much data and resources. For this issue we designed the game to have prompts that guide what the players draw to try to prevent this issue. Portability is an issue if a WebGL browser cannot run on a particular device, but we designed our system to be able to run on most desktops. Security can be an issue via possible cross-site scripting attacks since there being no text input validation and sanitization. For this issue we designed our system to save no personal information about our users such as having login credentials.

5c5 Prototypes to Evaluate Alternative Design Strategies

We will need to evaluate alternative design strategies with our initial prototype, which is the 0.1 prototype.

5c6 Expected Technical Difficulties

The technical difficulty we expect to encounter is an interfacing problem with an IOS device that uses Safari as a web browser. We will solve this technical difficulty by changing the format from WebGL to other technology for this circumstance.

5d Design Viewpoint 2

Lower-Level Architectural Design Viewpoint.

5e Design View 2

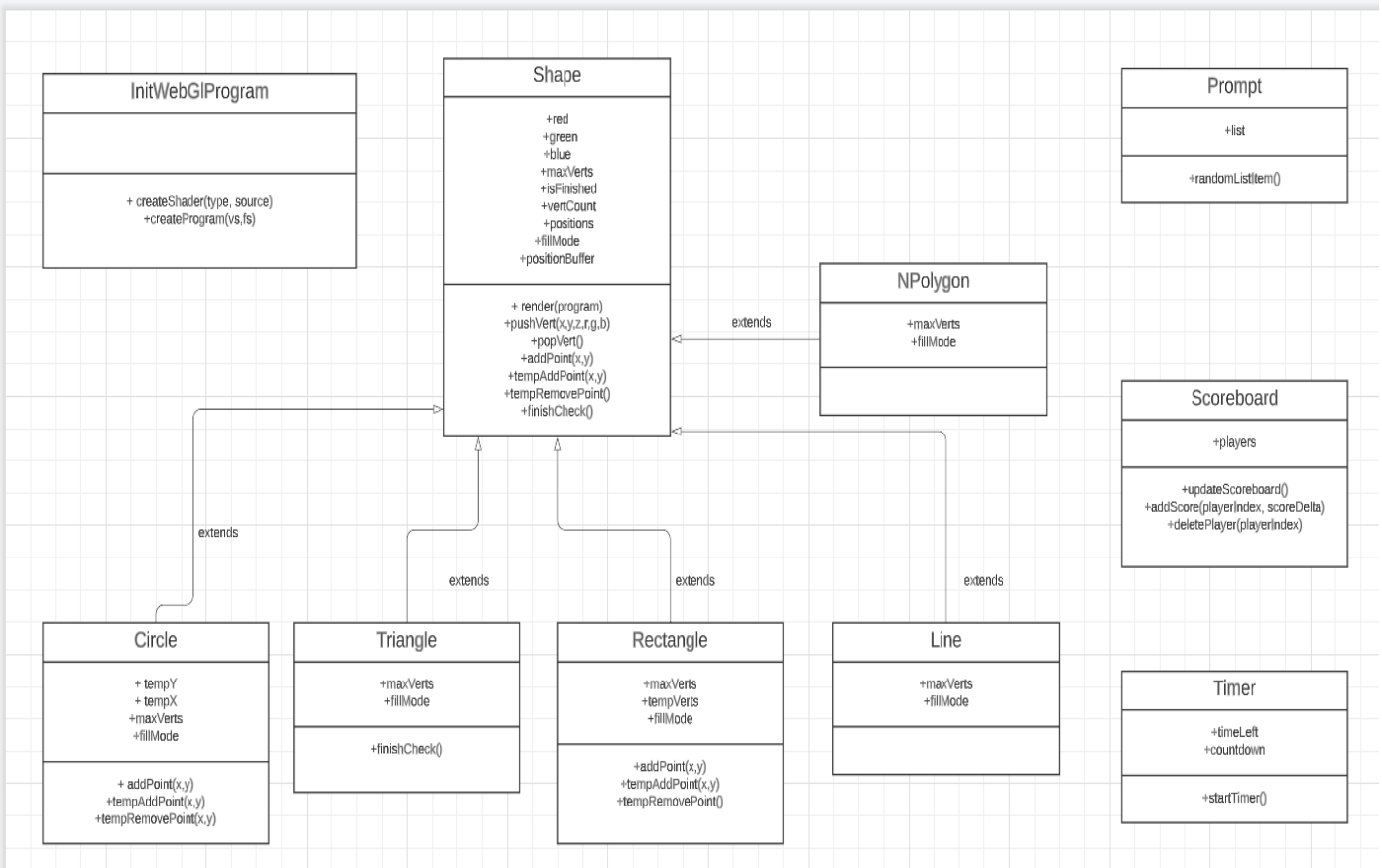

Figure 5 - UML Diagram

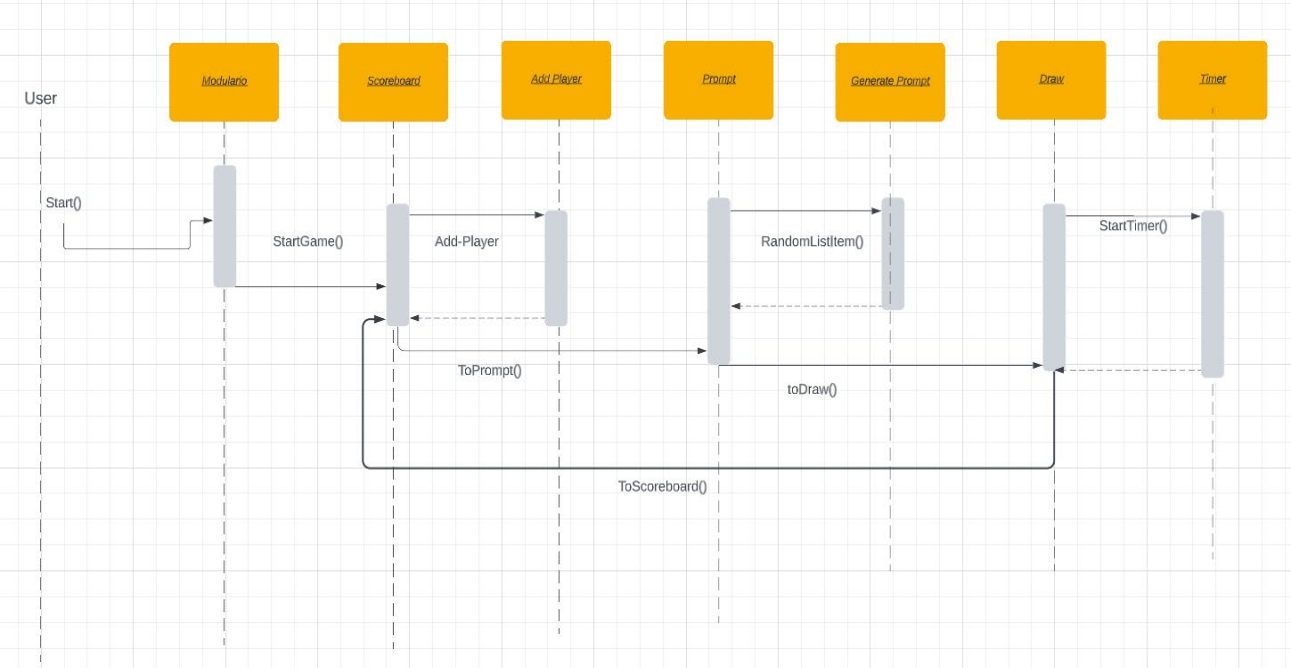

Figure 6 - Sequence Diagram

5e1 Interfaces, Preconditions, and Postconditions

Interfaces

- Main U.I. - composed of color and shape selection.
- Keyboard
- Mouse
- All API-type interfacing will happen in a single JS program embedded in the site's HTML.

Classes

- InitWebGL - Starts an instance of WebGL, the main API, to be used by our program.
    - Preconditions - None; this class is built to fulfill preconditions for other classes.
    - Postconditions - Must Generate at least one compiled WebGL program at runtime; the program will be composed of 2 shaders, a fragment shader, and a vertex shader, used for color and positioning respectively.
- Shape - An abstract class used to hold the significant rendering behavior and other abstractable concepts related to the position and vertice count of other shapes.
    - Preconditions - Must have a working WebGL program running in the environment for the shape to latch onto and render, must be fed color information.
    - Postconditions - Must have color, vertice number, and vertice position established by the end of its rendering call.

- Circle - a concrete class, most behavior is inherited from the shape class.
    - Preconditions - Must have a fill and color fed into its constructor. The mouse must have made a click recently that was recorded and stored as a position.
    - Postconditions - Must define & display a finite space of a circle that has 20 distinct points along it that are mathematically defined.
- Line - A concrete class, most behavior is inherited from the shape class.
    - Preconditions - Must have a fill and color fed into its constructor. The mouse must have made a click recently that was recorded and stored as a position.
    - Postconditions - Must define & display a line between 2 points that is a color that the user has selected.
- Triangle - A concrete class, most behavior is inherited from the shape class.
    - Preconditions - Must have a fill and color fed into its constructor. The mouse must have made a click recently that was recorded and stored as a position.
    - Postconditions - Must define & display a triangle with 3 points and a color; if filled fill color must match the outline color.
- Rectangle - a concrete class, most behavior is inherited from the shape class.
    - Preconditions - Must have a fill and color fed into its constructor. The mouse must have made a click recently that was recorded and stored as a position.
    - Postconditions - Must define & display a Rectangle with 4 points and a color; if filled fill color must match the outline color.
- NPolygon - a concrete class, most behavior is inherited from the shape class.
    - Preconditions - Must have a fill and color fed into its constructor. The mouse must have made a click recently that was recorded and stored as a position.
    - Postconditions - Must define & display a Npolygon with N points and a color; if filled fill color must match the outline color.
- Main - contains the main execution class, tells all shapes in its arrays when to render.
    - Preconditions - N/A
    - Postconditions - program will be finished running.

- Scoreboard - A class that contains a single int score that will.
    - Preconditions - N/A
    - Postconditions - Final score will be decided.

Functions

- addPoint(x,y) - belongs to the shape family of classes and is defined by the class that inherits from Shape, will decide whether we can add a point then add a point to the positions array if it can.
    - Preconditions - the shape to have a point added must have less than the maximum number of points for that individual shape.
    - Postconditions - a point will be added to positions, the vertice count for the shape will be incremented, the new point will then be buffered into WebGL .
- pushVert(x,y,z,r,g,b) - belongs to the Shape class, typically called from addPoint after the number of points has been checked.
    - Preconditions - must be given a valid xyz point and a valid rgb color.
    - Postconditions - will add a specified point with specified color to the positions array of the shape unconditionally.
- popVert() - removes a single vertice from the positions array.
    - Preconditions - at least one vertice in the positions array.
    - Postconditions - one less vertice in the positions array.
- tempAddPoint(x,y) - adds a temporary point at render time to allow a ghost to render following the mouse but prevent permanent addition to the positions array.
    - Preconditions - the shape must not be finished.
    - Postconditions - adds a single temporary vertice to the positions array.
- tempRemovePoint() - removes the temporary point so we don't consider it as permanent during calculations.
    - Preconditions - must have at least one point that is temporary in the positions array.
    - Postconditions - will remove the last added temporary point.
- finishCheck() - checks whether it should be finished and changes fill mode to appropriate mode to match.
    - Preconditions - must be called on a shape.
    - Postconditions - set appropriate fill mode based on whether user is done editing current shape.
- createShader(type,source) - creates and compiles a GLSL shader given GLSL source and type of shader to be created.
    - Preconditions - needs shader source code to be input.

    - Postconditions - returns either a compiled shader or any compilation errors encountered.
- finishCheck() - checks whether it should be finished and changes fill mode to appropriate mode to match.
    - Preconditions - must be called on a shape.
    - Postconditions - set appropriate fill mode based on whether user is done editing current shape.
- createProgram(vs,fs) - creates a WebGL program given a compiled vertex shader and fragment shader.
    - Preconditions - need compiled shader that is error free.
    - Postconditions - returns either a compiled WebGL program or a compilation error that it encountered.

5f Design Rationale

Our goals of our web application drove our design decisions to focus on high-portability, quality, and ease of use for users. One design option we considered was having multiple users be able to join over an internet connection. We decided against this design option in order to set our web application apart from other applications by making it a local application.

IV Phase 4 – Testing and Final Product

1 Lesson Learned

We benefited from this project by learning, understanding, and applying the development of software from start to finish. In phase 1, the major topics we learned about were the software development life cycle, project team organization, and the different types of plans such as security plan, risk management plan, and maintenance plan. Many of these topics can be used for different types of projects, not only software, and forced us to consider not only the aspects of the project we know for a fact but also the possibilities of what could happen. In phase 2, the major topics we learned about were what a Software Requirements Specification looked like, functional and nonfunctional requirements, and the use case diagram. This phase demonstrated how diagrams can be useful in understanding a software process and how important documentation formatting is. Finally, in phase 3, the major topics we learned about were what a Software Design Specification looked like, dependencies and execution views, architectural styles, design issues, and design views. This phase showed the importance of the previous phases and how all the planning comes together to make the implementation in the next phase run smoother. The knowledge and tools that we take away from this project will benefit us in our future careers and group projects whether they are software related or not.

2 Screenshots of Program

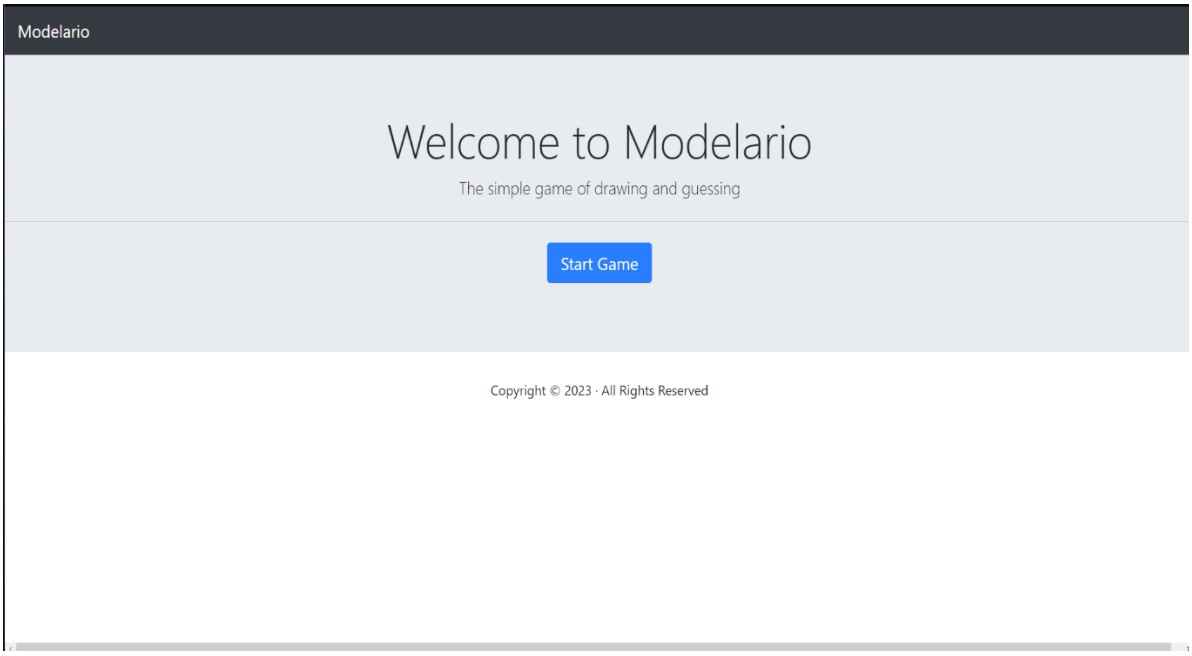

Figure 7 - Modelario Start Page

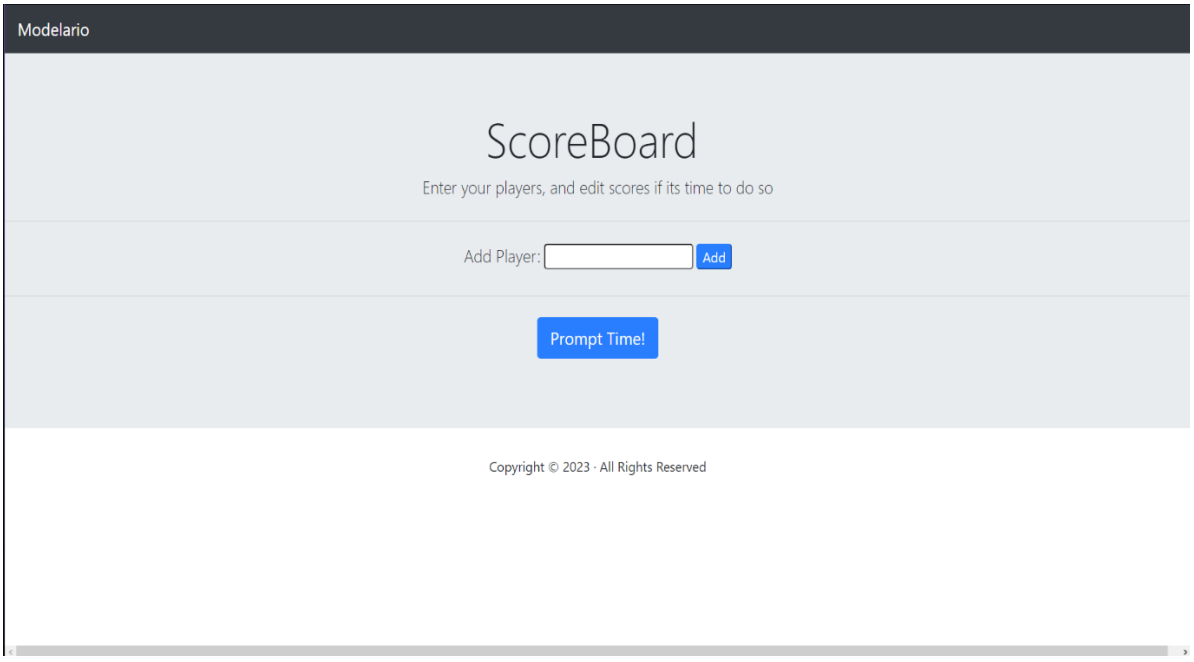

Figure 8 - Modelario Scoreboard Page

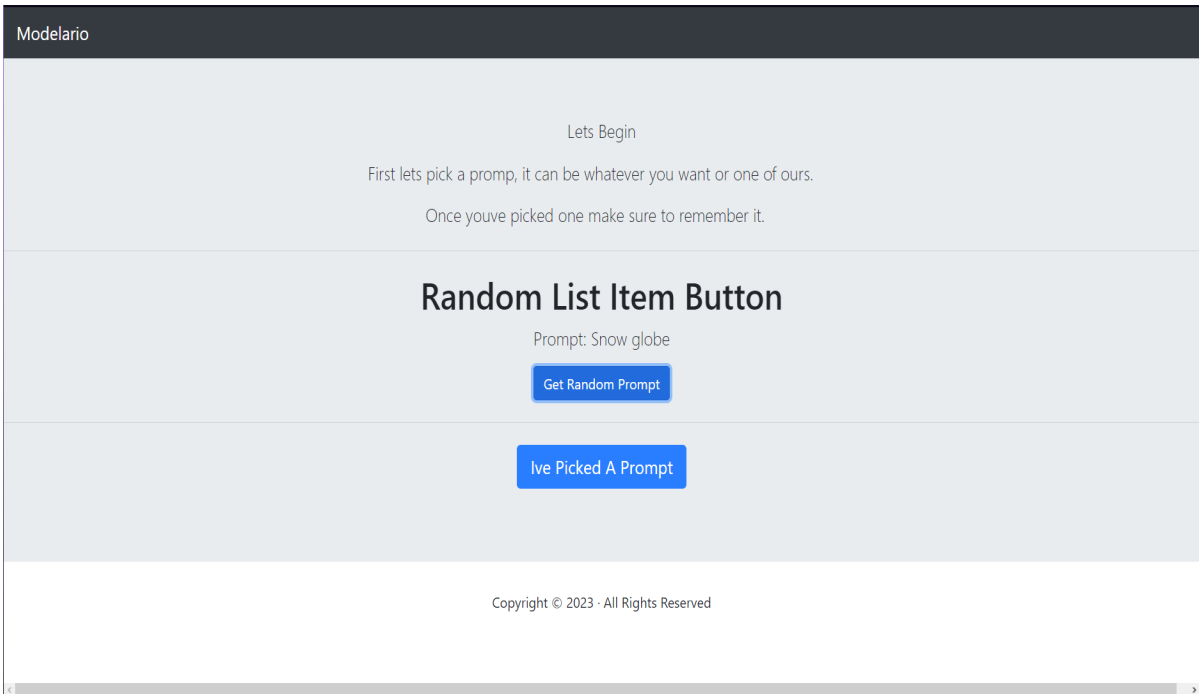

Figure 9 - Modelario Prompt Page

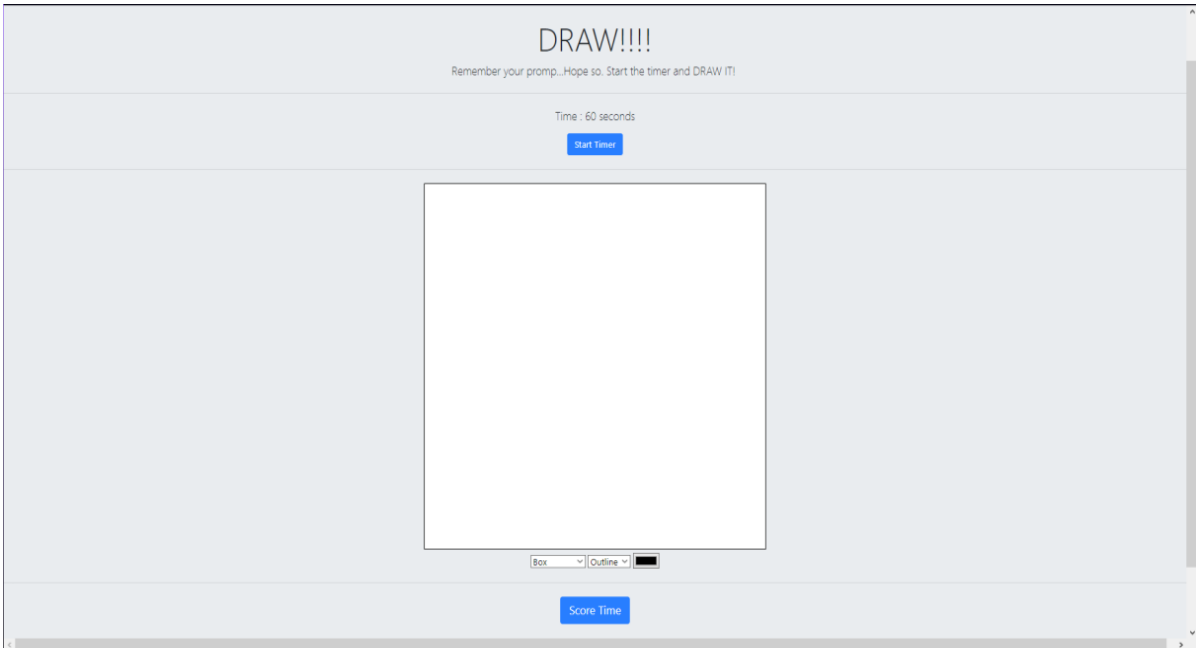

Figure 10 - Modelario Drawing Page

3 Actual Time Worked Compared to Initial Estimates

*Table 1 - This table calculates the difference between the actual hours worked vs the initial estimates for each task. Values with " * " symbol are milestones merged into one overall time.*

Actual Time Worked (Hours)		
Task	Actual vs Estimated	Difference
1.1	1 - 1	0
1.2	1 - 2.75	-1.75
1.3	1 - 2.5	-1.5 *
1.4	1 - 1	0 *
1.5	1.25 - 1.125	+.125
1.6	1.25 - 3.75	-2.5 *
1.7	1.25 - 4.75	-3.5 *
1.8	0 - 1	-1
1.9	0 - 2.5	-2.5
1.10	0 - 3	-3
2.1	1 - 2.75	-1.75
3.1	4 - 4	0
3.2	4 - 4	0 *
3.3	4 - 4	0 *
Total		-11.125

Significant differences came along the scoreboard and Canvas tool milestones (1.5-1.10). This is due to a safe overestimate of the time it would take to make the scoreboard, as well as us having the canvas drawing tool mostly functional going into the development phase. Most of the other estimates are relatively close to the actual time spent on said activity, with the testing phase actual time being identical to the estimate.

4 Time Sheet

Table 2 - This table shows the records of the tasks each team member completed, on what day, and how long it took them to complete the task.

Team Member Name	Task	Date	Time (minutes)
Justine	IEEE Standards	3/23	4:15-5:30 pm (75)
Justine	Document Prep	3/28	2:30-3:15 pm (45)
Justine	IEEE Template	4/10	9:20-10:00 pm (40)
Justine	IEEE Template	4/11	10:10-11:00 am (50)
Justine	Final Report	4/24	8:00-9:00 pm (60)
Justine	Final Report	4/25	6:00-9:00 pm (180)
Chase	IEEE Standards	3/23	4:15-5:30 pm (75)
Chase	Functional Requirements	3/28	4:50-5:00 pm (10)
Chase	Interface Graph	4/11	3:30-4:45 pm (75)
Chase	Interfaces, Classes, and Conditions	4/11	11:00-2:30 pm (150)
Brayden	UML Diagram	3/24	11:30-12:30 am (60)
Brayden	Functional Requirements	3/28	4:50-5:00 pm (10)
Brayden	UML Class Diagram	4/10	10:00-11:00 pm (60)
Brayden	Sequence Diagram	4/11	11:15-11:45 am (30)
Brayden	Revision of UML and Sequence Diagrams	4/25	7:00-8:00 pm (60)
Adrian	UML Diagram	3/24	11:30-12:30 am (60)
Adrian	Website Setup	3/30	3:40-4:40 pm (60)
Adrian	Major Components	4/6	5:00-6:00 pm (60)
Adrian	Scoreboard	4/11	3:30-4:45 pm (75)
Adrian	Module Integration	4/21	3:30-4:30 (60)
Adrian & Chase	Testing	4/21	4:30-5:30 (60)
ALL	Plan for Phase II Meeting	3/21	4:50-5:10 pm (20)
ALL	Wrapping Up Phase II Meeting	3/28	4:50-5:00 pm (10)
ALL	Plan for Phase III Meeting	4/4	4:50-5:10 pm (20)
ALL	Team Evaluation of Design Issues Meeting	4/6	4:50-5:20 pm (30)
ALL	Wrapping Up Phase III Meeting	4/11	4:50-6:00 pm (70)
ALL	Revisions	4/23	7:00-8:00 pm (60)

5 Testing Results

For testing, we implemented a variety of methods to verify both the functionality and integrity of the program. The first method we devised was unit testing. We ran through picking each function, compiled it into a list of functions, then ran each function with our black-boxed results in mind. These tests were entirely successful. We then ran some full system tests, pushing each component to and past a reasonable limit. The system passed these tests with flying colors, being able to have either infinite or 0 players and being able to store well over 300 shapes without hitching or frame drops. We also deployed some light integration testing, building some system components in isolation, namely the embedded drawing program, and tested it to make sure all functionality was present, which it was.