Project Management Plan

WebGL Card Games

Version 1.5

2/20/19

Team WebGL Game

https://docs.google.com/document/d/1cUiXDhSkCCtY6ENkNGBB8LzcF0NsvJ3VDmQUY-nPF4c/edit

Computer Science / Bellevue College

Sara Farag

Revisions

Version	Primary Author	Description of Version	Date completed
1.0	Sean Hardin, Anthony Klobas, Jeffrey Talada	Created primary version of document	10/2/18
1.1	Anthony Klobas, Sean Hardin, Jeffrey Talada	Updated literature review, updated gantt chart	10/17/18
1.2	Jeffrey Talada	Addressed issues with 1.1, updated gantt chart	11/20/18
1.3	Jeffrey Talada	Finalized for Milestone 1	12/4/18
1.4	Anthony Klobas, Jeffrey Talada	Updated Resources, Tasks, Time Table	2/8/19
1.5	Jeffrey Talada	Updated Tasks and Time Table	2/20/19

Table of Contents

1. Introduction

1.1 Project Overview

Purpose

Our purpose is to gain experience building an application in WebGL while working in an AGILE fashion.

Scope

We will create an online card gaming platform, starting with Blackjack, using WebGL and put it in a webpage using Javascript and Node.js. It will support multiplayer and single player versus an AI. In the future, we will extend the platform to include other functionalities, such as chat, more AI options such as difficulties, personalities, and strategies, and other game advanced options.

Assumptions and constraints

We will support 1-8 players depending on the game. We will assume that users of our game will have an up to date version of their web browser that supports WebGL (listed below)[1] and at least a 56kbps internet connection.

- Chrome 8+
- Firefox 4+
- Safari 5.1+
- Edge
- IE 11
- Opera 12.1+

1.2 Literature Review

- According to our research, the client-server model makes it easiest to guard against cheating. Little research has been done on maintaining security in P2P transactions and going with a server drastically reduces the difficulty of implementing secure game play.[2]
- Using Node.js for our backend will allow us to use the same code on both client and server (as both run javascript).[3]
- For the purposes of scaling we will break our script into various processes[4] because node.JS is single threaded.

[1] Caniuse.com, 'WebGL - 3D Canvas graphics'. [Online]. Available: https://caniuse.com/#feat=webgl. [Accessed: 20- Nov- 2018].

[2] C. Neumann et al., 'Challenges in Peer-to-Peer Gaming'. [Online]. Available: http://ccr.sigcomm.org/online/files/p2p_gaming.pdf. [Accessed: 20- Nov- 2018].

[3] J. Rachowicz, 'When, How And Why Use Node.js as Your Backend', 2017. [Blog]. Available: https://www.netguru.co/blog/use-node-js-backend. [Accessed: 20- Nov- 2018].

[4] R. Jahchan, 'Taking Advantage of Multi-Processor Environments in Node.js', 2014. [Blog]. Available: https://blog.carbonfive.com/2014/02/28/taking-advantage-of-multi-processor-environments-in-node-js/. [Accessed: 20- Nov- 2018].

- Our system will be broken up into a main "server" section that handles authentication and match making, but have clients passed to sub components for the purposes of a single game.[5]
- Making use of ssl encryption will help prevent malicious sites from impersonating us, or bad actors from "faking" games to scam people out of money.[6]
- We will use Jquery because it makes manipulating the DOM easier and helps with asynchronous resource requests.[7] The amount of DOM manipulation we need is minimal so React would unnecessarily bloat the RAM usage of our application.[8]
- We've chosen JSON over xml because it can easily be parsed directly into javascript objects while xml would require processing.[9]
- We've chosen to use AzureDevOps because it supports both project management tools with automatic charting and metrics as well as CI/CD for project development.[10]
- For safety we're not storing passwords but instead hashing them in case the database leakes. From what we've read bcrypt is a pretty safe hashing tool to use[11]

[5] M. Mecinski, 'Architecture of a Node.js multiplayer game', 2017. [Blog]. Available: https://medium.com/@MichalMecinski/architecture-of-a-node-js-multiplayer-game-a9365356cb9. [Accessed: 20- Nov- 2018].

[6] D. Beattie, 'Why Use SSL/TLS? 4 Reasons besides Encryption', 2016. [Blog]. Available: https://www.globalsign.com/en/blog/why-use-ssl-besides-encryption/. [Accessed: 20- Nov- 2018].

[7] O. Akinseye, 'jQuery vs Vanilla JavaScript - Deciding on What to Use', 2017. [Blog]. Available: https://www.codementor.io/brainyfarm/jquery-vs-vanilla-javascript-deciding-on-what-to-use-6b79xdmrv. [Accessed: 20- Nov- 2018].

[8] Reactjs.org, 'Virtual DOM and Internals'. [Online]. Available: https://reactjs.org/docs/faq-internals.html. [Accessed: 20- Nov- 2018].

[9] w3schools.com, 'JSON vs XML'. [Online]. Available: https://www.w3schools.com/js/js_json_xml.asp. [Accessed: 20- Nov- 2018].

[10] Microsoft, 'Compare features between plans'. [Online]. Available: https://azure.microsoft.com/en-us/services/devops/compare-features/. [Accessed: 20- Nov- 2018].

[11] Lucas kauffman 'About Secure Password Hashing' [blog]. Avaliable: https://security.blogoverflow.com/2013/09/about-secure-password-hashing/

2. Project Organization

2.1 Roles and Responsibilities

Team Member	Roles	Email
Sean Hardin	Developer	sean.hardin @bellevuecollege.edu
Anthony Klobas	Developer	aklobas @bellevuecollege.edu
Jeffrey Talada	Team Leader, Developer	j.talada @bellevuecollege.edu

2.2 Tools and Techniques

- Azure DevOps - Project management, hosting, CI/CD, code repository
- Bootstrap - Prebuilt open source responsive websites
- Browser Suite - Deployment Testing (Chrome, Firefox, Edge, Safari)
- Bcrypt - hashing passwords
- Express - static http server
- MySQL - Database (more powerful, future deployment)
- Sqlite3 - Database (smaller, faster for prototyping)
- Workbench - MySQL IDE
- Node.js - Backend
- Nodemon - development/rapid prototyping
- Three.js
- jQuery
- JSON
- Javascript - Frontend
- HTML/CSS - Webpage
- WebGL - Application graphics
- Discord - Communication
- Docker - Development and deployment

3. Project Management Plan

3.1 Tasks

Task#/Feature#	Task Description	Date
T1	Project Management Plan	10/3/18
T2	Play with WebGL and make a 2D square	10/12/18
T3	Assign user stories to break into tasks	10/5/18
T4	Set up website	10/12/18
T5	Software Requirements Document	10/10/18
T6	Break user stories into tasks	10/15/18
T7	Software Design Document	10/24/18
T7a	Sections 1 and 2	10/24/18
T7b	Section 3	10/24/18
T7c	Section 4	10/24/18
T8	Make a 3D scene in WebGL	10/19/18
T9	Make rotating cube	10/26/18
T10	Software Testing Document	11/21/18
T10a	Choose Javascript Unit Testing Framework	11/19/18
T11	Prepare Presentation on OpenGL Pipeline	11/14/18
T12	Update and Fix PMP	11/28/18
T13	Get Azure Pipeline Going	11/28/18
T14	Review, Update, and Submit all Documents	11/28/18
T15	Prepare a Presentation on Project	11/28/18
F1	Client code	11/28/18
F1T1	Make basic controller and model, GLObject related classes, and JSON template	11/28/18
F1T2	Make renderer, scene, and view classes	11/28/18

T16	Refactor View	Sprint 5
T17	Make Spread (Bezier)	Sprint 5
T18	Client Connection and Game Instantiation	Sprint 5
T19	Blackjack Dealer AI	Sprint 5
T20	Blackjack Model	Sprint 5
T21	Turn Taking	Sprint 5
T22	Update Documentation	Sprint 5
T23	User Login	Sprint 6
T24	Game Selection	Sprint 6
T25	Blackjack Player AI	Sprint 6
T26	Put Card Textures into a single image	Sprint 6
T27	Dynamic Buttons	Sprint 6
T28	Player Summary	Sprint 6
T29	Update Documentation	Sprint 6
T30	Beautify	Sprint 7
T31	Credit Transfer	Sprint 7
T32	Settings Screen	Sprint 7
T33	Default add AI players	Sprint 7
T34	Add betting, splitting, double, surrender to blackjack model	Sprint 7
T35	Handle Dropping Players	Sprint 7
T36	Update Documentation	Sprint 7
T37	Texas Hold’em Model	Sprint 8
T38	Decouple Client and Model	Sprint 8
T39	Table Selection	Sprint 8
T40	Node.js Presentation	Sprint 8
T41	Update Documentation	Sprint 8

T42	Winter Milestone Presentation	March W2
F4	Add Telemetry	End of Spring Quarter
F5	Make Mobile	End of Spring Quarter
F6	Add ADA compatibilities or other	End of Spring Quarter
T43	Final Presentation	End of Spring Quarter

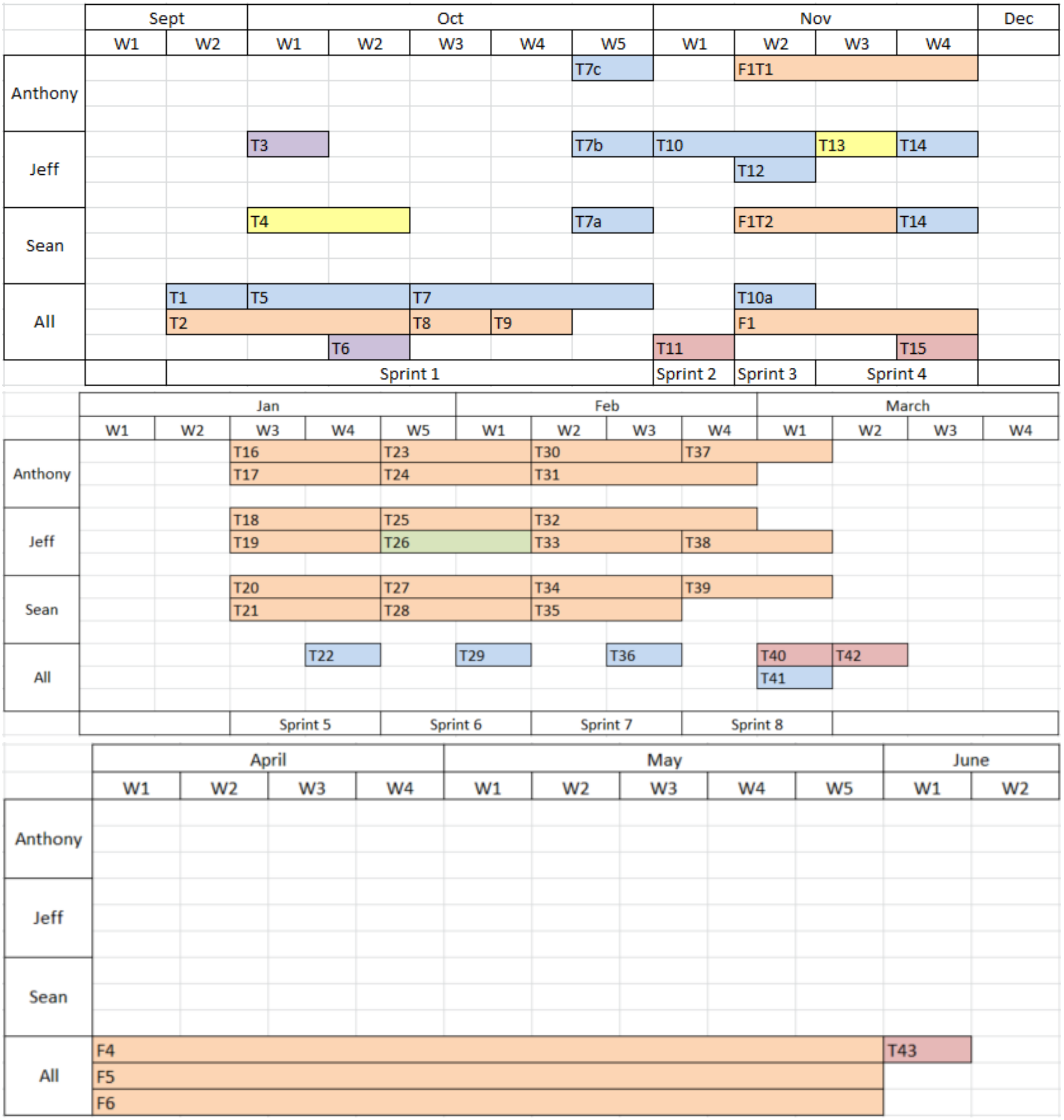
3.2 Assignments

Week#	Deliverables/Progress
Fall-3	Project Management Plan
Fall-5	Requirements Document
Fall-7	Development/source code
Fall-8	Design Document, Development/source code
Fall-9	Development/source code
Fall-10	Development/source code
Fall-11	Presentation + code + final versions of all documents
Winter-4	Development/Documents
Winter-6	Development/Documents
Winter-8	Development/Documents
Winter-10	Development/Documents/Tech Talk
Winter-11	Milestone Presentation

3.3 Timetable

- Blue - Documentation
- Orange - Coding

- Purple - Project Management
- Yellow - Infrastructure
- Red - Presentation
- Green - Assets



	Sept		Oct					Nov				Dec
	W1	W2	W1	W2	W3	W4	W5	W1	W2	W3	W4	
Anthony							T7c		F1T1			
Jeff			T3				T7b	T10		T13	T14	
									T12			
Sean			T4				T7a		F1T2		T14	
All		T1	T5		T7				T10a			
		T2			T8	T9			F1			
				T6				T11			T15	
		Sprint 1						Sprint 2	Sprint 3	Sprint 4		

	Jan					Feb				March			
	W1	W2	W3	W4	W5	W1	W2	W3	W4	W1	W2	W3	W4
Anthony			T16		T23		T30		T37				
			T17		T24		T31						
Jeff			T18		T25		T32						
			T19		T26		T33		T38				
Sean			T20		T27		T34		T39				
			T21		T28		T35						
All				T22		T29		T36		T40	T42		
										T41			
			Sprint 5		Sprint 6		Sprint 7		Sprint 8				

	April				May					June	
	W1	W2	W3	W4	W1	W2	W3	W4	W5	W1	W2
Anthony											
Jeff											
Sean											
All	F4									T43	
	F5										
	F6										

Additional Material

Definitions, Acronyms and Abbreviations

- AGILE - an approach to software development that emphasizes “adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.”[12]
- AI - artificial intelligence
- CD - Continuous Deployment
- CI - Continuous Integration
- IDE - Integrated Development Environment
- JS - Javascript
- JSON - Javascript Object Notation

Appendices

References

[12] Agile Alliance, ‘Agile 101’, 2018. [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed: 20- Nov- 2018].