98Point6 Interview Software Design Document

Stan Rosenbaum - Jan 2021

Reviewers: 98Point6 Hiring Committee

# INTRODUCTION

## Document Outline/ Overview of Contents of Document

This document is a blueprint for the architecture for the 9dt mobile game for iOS. This document will begin with a cover a high level description followed by lower level description to provide details for the application system.

## Goals

The purpose of the product to provide simple light diversion, in the form of a game in which the user can play is quick short bursts.

## Scope

This document is over covering the high and low abstract descriptions of the needed designs, data structure, and algorithms to adequately describe the requirements that must be met to completion of functional project. Actual internal implementation details are left to the developments team(s) to provide fallibility for day-to-day tasks and goal. This document is to provide a guide to the an high-level of the internal operation and presentation processes of the application, as well as recommended stubs that correspond to those outcomes.

## System Overview

The target deploying platform is any and all iOS and iPadOS platforms running iOS/iPadOS version 12 or higher

This is makes the earliest hardware supported

* iPhone 6s (2015)
* iPhone 6s Plus (2015)
* iPhone SE (2016)
* iPad mini 4 (2015)
* iPad Air 2 (2014)
* iPad 5 (2017)

Any models of hard older than these device are considered obsolete.

For development, each developer will require a MacOS desktop, capable of running XCode 10.0 or greater at a minimum. As development without Apple Branded product is more trouble than it is worth.

## Milestones

Milestones are unknown at this time, but givens a hypothetical development time of one month, we are planning for the midpoint review two weeks in and final review at the end of the month.

# General Overview and Design Guidelines/Approach

# Assumptions/Constraints/Risks

## Assumptions

We are assuming the game server is operating under the same specification as detailed in the proposal document.

Application should be easiely modifiable and scaleable for future versions. To try size sized grids.

## Constraints

We must ensure that the application has internet connectivy .

We also must ensure the interface is accessible

## Risks

Time is always the number one concern. Although a Month development should be ample time to allow for simple project

Team size. Due to the small size and scope of the project, team size can be a concern. This project should not require a large or even a medium sized development teams. One + one developer, a designer, and a project manager should be enough to cover most cases, where the designer or one of the development may even be able to double as project manager. Concern must be take to make sure the team does nto become too large, as then it would become a text book case of wasted resources, or in classical mythical man month an impediment to the project itself.

# Design Considerations

## Goals and Guidelines

Make efficient use of the UIKit for the presentation.

Use of external assets to minimize use of custom UIViews.

Development of the clean and easily understandable code flow that the project could be used in middle school instruction and understood by students. Like standard English prose, the team should aspire to write code easily understood able at a 5th grade level to achieve maximum readable and understanding.

Clear cut use of the Model View Controller (MVC) Patterns for code reuse if ported to MacOS platforms.

## Development Methods & Contingencies

## Performance Engineering

## Assumptions and Dependencies

We are assuming the game’s REST API server is always available to provide turn responses.

## General Constraints

Time – No More than a month.

Apple Store – Approval

Server – Reliability

## Goals and Guidelines

The game internal data structure and UI presentation should be develop as to be easily scalable as possible for future change.

The interface should be simple clean, accessible (for Voice Over and other motor impairments).

Localized, so it will be localizable ready, to expand into other language markets.

## Requirments

1. The app must allow the player to choose whether they want to go first, or if they want our service to go first.
2. If there is a win on either side, the app must display who won and let the player play again.
3. If the board is full, the app must tell the user the game is a draw and let the player play again.

## Development Methods

A general Agile Scrum methodology, with a planned technical debt time per sprint.

Although Test Driven Development can be uses to verify the integrity of parts of the system, the bulk of testing must be active user testing due to the interactive nature of the application as a game.

General code reviews should be encouraged. If possible, the 5th grade legibility test should be used toward the end of the project.

## Risks and Alternatives

Time is always a concern.

Data formatting from server is not normal JSON data, this prevents use of Codable data structures to simplify data.

Game state is checked and entirely maintained by client app. Automated tested with the server, although possible, is not guaranteed to return consistent results.

# Proposed Solution

# Architectural Design

## Client

This is a client application that communicates with a Stateless Anonymous Server.

Client / Server Software Layers

# Component design

## Application Overview

UIKit Single Window, Single Scene Application

Use of OS.Logger to increase debug reporting

Unit Tests whenever possible to test individual components.

## Human User Interface Design and 508 Compliance

Being an interaction application, we are addressing.

### General UI Layout

A picture containing icon

Description automatically generated

### Detailed Break Down

A picture containing logo

Description automatically generated

#### A: Single GameCell

This UI Element, should be based off of UIKit’s UIImageView element. This element represents a single token in the game grid. UIImageView is chosen because the token avatar can be represented by a set of images and identifies by image literals in the Swift language. Similarly, The set of images representing a range of token can be store in a XCode Asset Library.

. The Images for each cell should be set programmatically. If possible, use PDF vector files for the default image format as XCode can automatically generated 1x, 2x, 3x… Similarity, all images access should be places in a dedicated asset catalog file.

Where individual cells should be accessibly remains to be determined (see later in document).

#### B: Single Game Column

A Game Column, represents a set of game cells stacked vertically.

Added tokens should fill the bottom most empty slot abailable in the column until the column is full.

The set of game cells should be stored in vertical StackView,

This should allow the base of the UICollection Content Cell which makes up the game field.

The Column should be accessible as it represents the lowest interactive

C: The Game Field.

The game field is made up a UICollectionView in a fixed Grid view of one 1 row and X number of Columns. In this case 4 columns.

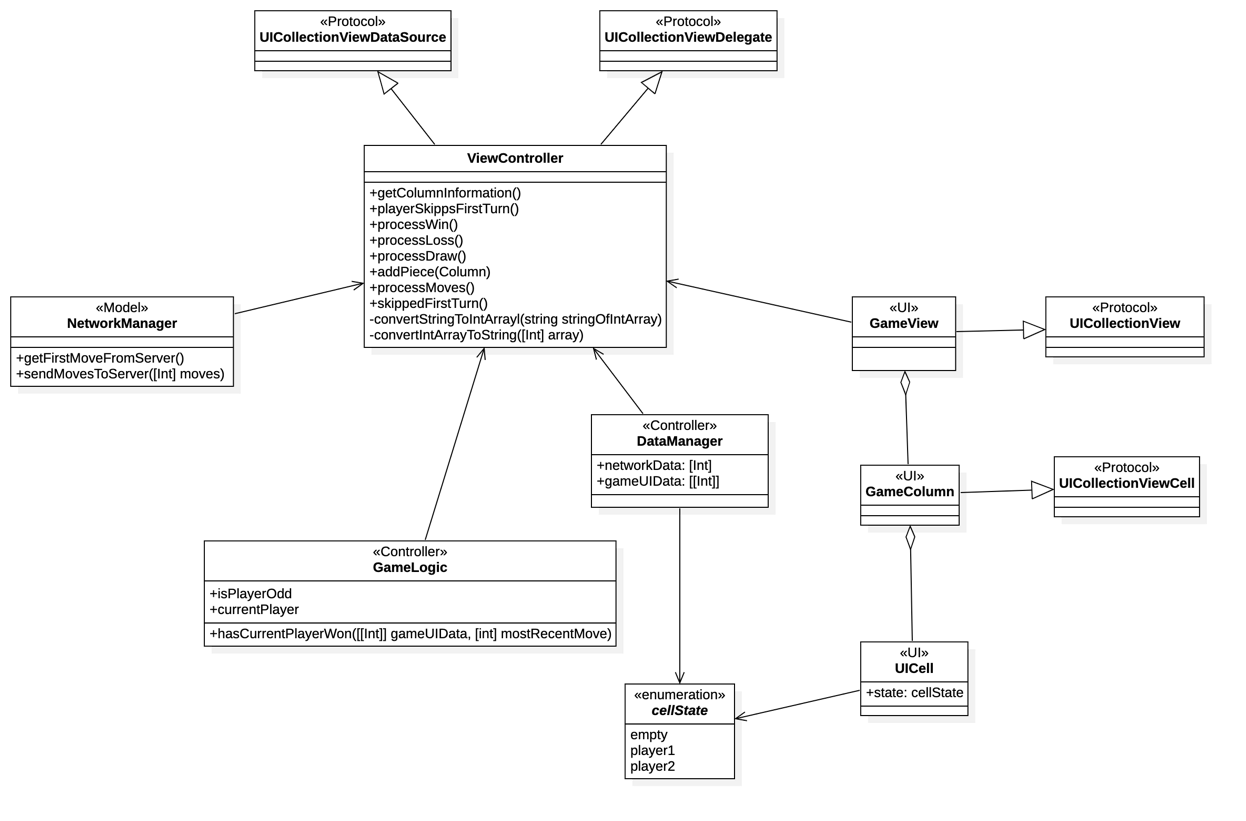
Using a collection view in this way will allow the game field to be extenable in future iterations.

D: Information / Pass Button.

This is just a simple button with label that can be used to convey information to the user as well to confirm basics interactions with the user.

## System Architecture

### General Overview



#### Description

Standard Model View Controller (MVC)

The App should use a standard MVC model where their Controller is the central point for

##### Model

NetworkManager(): Application’s contact point with the server via network URL calls.

##### Controller

ViewController():

The ViewController is the central hub for the app. It should responsible for the central data flow and delegation of processes. For example, it should handle the conversion of the server’s string response to an Integer Array and back for the DataManager. However, the DataManager handles the processing of the server data to a representation needed for the displaying UI elements and game logic. Since the UI makes use of collection view, this class could be a UICollectionViewController[[1]](#footnote-1).

DataManager():

The DataManager job is to negotiate maintain the current game field data in the form of a fixed array of arrays, columns first.

We want to use columns first because that can simply report column state for Accessibility announcements.

By using an array of array, game win and loss then be calculated recursively (similar to the N-Queen problem, see GameLogic for more).

GameLogic()

The primary job of the GameLogic Class is to determine if there is a winner with every added turn. The GameLogic classes check if the most recent piece added results in a winning condition.

##### UI

GameView

The GameView should be 2 dimensional collection of gameCell views grouped in by columns first represented by UICollectionView[[2]](#footnote-2). Columns is preferred because it organize the game state information into a form that can quickly be reference by UI and Accessibly VoiceOver to pass information on to the User.

Related: The columns should be interacted with as one whole unit. The Accessibility label for each column unit will need to be set to read the content of the column from the bottom up. Therefore, as the smallest accessible unit each column should be IDs as the UICollectionViewCell[[3]](#footnote-3).

Accessibility Magic Tap for each column triggers should also trigger the game to respond as if the player selected the column for their turn.

GameCell:

The individual cells in each column that makes up the UICollectionViewCells should be UIImageView [[4]](#footnote-4)component.

Whether GameCell should have accessible is unknown and this could be revised after user testing. However currently the UIImageView accessible should currently be disabled, since it is the content of the columns as a whole which is the primary information.

## Logical and Process Flows

### General Game Logic (High Level)

A picture containing text, parking

Description automatically generated

This is a high-level view of the general logic flow for the game.

Notice, there is not a true decline state once the game is over. The game just remains in an ideal state until decides to play again.

## Component Breakdown and Analysis

### Internal Core Data Flow (High Level)

Graphical user interface, application, Teams

Description automatically generated

## Controller Data Flow Processing

Graphical user interface

Description automatically generated with medium confidence

### Network Processing Data Flow

Diagram, engineering drawing

Description automatically generated

## Security Architecture

No security architecture.

The largest security concern could be man in the middle attacks, of interception data between client and server.

Invalid data not applicable for the current game could be simply ignored.

Valid data, but non-authentic data could be used in liu of authentic data.

## Performance

No outstanding performance concerns unless the future design of the application ask the game field to be scaled up to the extraordinary sizes. If that that is the case the server REST URL Point will need to be redesigned before so before we can adaprt the client app.

The game should be providing good performance in terms of maintaining User engagement. Although,

# Data Design

## Component Design

## Data Objects and Resultant Data Structures

The software needs three primary data structures.

### DataManager

The DataManager needs to maintain the following.

1. A single array of integers, to communicate with the game server
2. A 2-dimensional array of integer, also known as an array of an array of integers. This is to hold a representation of the game data for the UI and for process the game logic for win states.

### Enums

One enum with the states of:

* Empty
* Player\_1
* Player\_2

This is to helps identify the states of the game cells.

## Data Conversion

The two areas of data conversion will happen in the Controller, and DataManager.

Controller needs to convert an integer array from string to Integer Array and back for communication with network URL calls.

DataManager must handle conversion internally from Integer Array to Multidimensional Integer array for game state representation.

Detailed System Design

Responsibilities

Constraints

Composition

Uses/Interactions

Resources

Processing

Glossary

# Analysis

## Alternative Solutions

## Open Questions

Performance analysis

Feasibility and resource estimates

Appendix A: Record of Changes

Appendix B: Acronyms 0

Appendix C: Glossary 1

# Appendix D: Reference Documents

These are document either referenced in this document or may provide addition information for specific areas.

## Asset Catalog

<https://www.simpleswiftguide.com/how-to-add-image-to-xcode-project-in-swiftui/>

## Collection Views:

<https://developer.apple.com/documentation/uikit/views_and_controls/collection_views>

## Localization

<https://medium.com/swlh/app-localization-in-swift-ios-swift-guide-baa2c2e4298e>

<https://medium.com/@Jeehut/localization-in-swift-like-a-pro-48164203afe2>

## Vector Images

[https://useyourloaf.com/blog/xcode-9-vector-images/](https://useyourloaf.com/blog/xcode-9-vector-images/ )

## Image Literals

<https://medium.com/ios-os-x-development/be-literal-76e9b4389eda>

<https://medium.com/@gurdeep060289/color-image-new-literals-in-the-cocoa-town-7ef4f2710194>

Appendix E: Approvals 3

Appendix F: Additional Appendices 4

Appendix G: Notes to the Author/Template Instructions 5

Appendix H: XLC Template Revision History 6

SOFTWARE REQUIREMENTS TRACEABILITY MATRIX

APPENDIX A: ITMS MODULE MAP 8

APPENDIX B: REQUIREMENTS TRACEABILITY MATRIX 9

Appendix C: Data Flow Diagram 0

APPENDIX D: APPROVALS 2

1. https://developer.apple.com/documentation/uikit/uicollectionviewcontroller?language=occ [↑](#footnote-ref-1)
2. https://developer.apple.com/documentation/uikit/views\_and\_controls/collection\_views [↑](#footnote-ref-2)
3. https://developer.apple.com/documentation/uikit/uicollectionviewcell [↑](#footnote-ref-3)
4. https://developer.apple.com/documentation/uikit/uiimageview [↑](#footnote-ref-4)