BetBuddy

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 19/10/2022 | 1.0 | First Version | Aaron Reith |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 4

5. Logical View 4

5.1 Overview 4

5.2 Architecturally Significant Design Packages 5

6. Interface Description 6

7. Size and Performance 6

8. Quality 6

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

## Scope

This Software Architecture Document provides an architectural overview of BetBuddy. BetBuddy is a mobile app that provides a log of bet history and informational graphs created using the extrapolated data to advise the customer of their betting tendencies and financial returns, which equips the user with the knowledge needed to create and maintain a stable, profitable return on investment.

## Definitions, Acronyms, and Abbreviations

BB: BetBuddy

## References

None.

# Architectural Representation

This document presents the architectural as a series of views; use case view, process view,   
deployment view, and implementation view. These views are presented as Rational Rose Models and use the Unified Modeling Language (UML).

# Architectural Goals and Constraints

BetBuddy is a stand-alone application held solely on the user’s cell phone, which includes all storage and no internet connection required.

# Use-Case View

The Use Case View is an important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

## Use-Case Realizations

Refer Use Case Realization document

# Logical View

This section describes the architecturally significant parts of the design model, such as its   
decomposition into subsystems and packages.

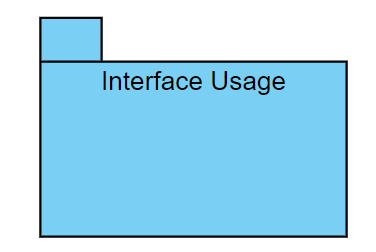
## Overview

This subsection describes visually the overall decomposition of the design model in terms of its   
package hierarchy and layers.

## Architecturally Significant Design Packages

### Interface Usage

As the sole package currently in the use-case model, *Interface Usage* manages both the interactions of the user and the system with the interface through the various use cases



**Figure 1 – Interface Usage Package Diagram**

|  |  |
| --- | --- |
| Property | Description |
| Name | Menu |
| Description | Responsible for all beginning interactions with the user |
| Methods | getBetHistory()  getViewStats() |
| Attributes | System: Object |

|  |  |
| --- | --- |
| Property | Description |
| Name | System |
| Description | Stores all essential data relating to bets |
| Methods | readBet() |
| Attributes | Bet: Object: |

|  |  |
| --- | --- |
| Property | Description |
| Name | Bet |
| Description | Outlines the Bet Object |
| Methods | getDate()  getSport()  getBuyIn()  getPayout() |
| Attributes | Date  Sport  BuyIn  Payout |

|  |  |
| --- | --- |
| Property | Description |
| Name | BetHistory |
| Description | Allows for interaction with past data |
| Methods | addNew()  edit()  delete() |
| Attributes | System: Object |

|  |  |
| --- | --- |
| Property | Description |
| Name | ViewStats |
| Description | Converts Bet data to Statistics that would be of value to the user |
| Methods | getStats()  getGraph()  getProfile() |
| Attributes | System: Object |

# Interface Description

See User Interface documents

# Size and Performance

The selected architecture supports the sizing and timing requirements through the implementation of the application. The client portion is implemented to ensure that minimal disk and memory requirements are   
needed on the android devise.

# Quality

The software architecture supports the quality requirements, as stipulated in the Software   
Requirements Specification and Supplementary Specification.