# Design Documentation – Matched Bet App

The application will be designed targeting users who place match bets using a mobile device. The application will include a matched bet calculator with basic calculations to calculate lay stakes, a bet finding tool to connect with a remote API and obtain real time odds data. Matched bets will be able to be saved on the device allowing users to easily view previously calculated matched bets. Users will also be able to set reminder notifications for bets.

Application will be designed adhering to the MVVM design pattern, utilising fragments, viewmodels and repository. The repository will provide an access point for the viewmodels to send/receive to/from the database and network API.

The application will be designed to meet the requirements of the Udacity Nanodegree Capstone project. However, after completion of the Nanodegree, the application will be further developed and uploaded to Google play store.

Further development of app after capstone project:

* Implementation of more sports, leagues and markets
* Implementation of filters to find bet searches
* Explore more API sources for odds. More bookmakers etc.
* Implement more calculators
* Notification for change of odds

Project start - 5th December

Planning and find best API – 5th - 7th December

Implement calculator and main structure of app – 10th December

Implement Room database and save functionality – 12 December

Connect to Odds API , retrieve and parse data – 15th December

Notifications – 18th December

Motion layout implementation - 22nd December

Finalise project – 27th December

# Rubric Specifications

Android UI/UX

| CRITERIA | MEETS SPECIFICATIONS |
| --- | --- |
| Build a navigable interface consisting of multiple screens of functionality and data. | Application includes at least three screens with distinct features using either the Android Navigation Controller or Explicit Intents.  The Navigation Controller is used for Fragment-based navigation and intents are utilized for Activity-based navigation.  An application bundle is built to store data passed between Fragments and Activities.  The application consists of 4 fragments and navigation between fragments is performed using the navigation controller. The user can navigate between fragments via the menu toolbar. Application bundle is built and data is passed between fragments as a bundle. Data is passed to the notification builder as a bundle in the pending intent. |
| Construct interfaces that adhere to Android standards and display appropriately on screens of different size and resolution. | Application UI effectively utilizes ConstraintLayout to arrange UI elements effectivelyand efficiently across application features, avoiding nesting layouts and maintaining a flat UI structure where possible.  Data collections are displayed effectively, taking advantage of visual hierarchy and arrangement to display data in an easily consumable format.  Resources are stored appropriately using the internal res directory to store data in appropriate locations including string\* values, drawables, colors, dimensions, and more.  Every element within ConstraintLayout should include the id field and at least 1 vertical constraint.  Data collections should be loaded into the application using ViewHolder pattern and appropriate View, such as RecyclerView.  The four fragments are designed using constrain layouts. The user interface is designed to be suitable for orientation changes and devices of different sizes. Resources are stored appropriately in the res directory. |
| Animate UI components to better utilize screen real estate and create engaging content. | Application contains at least 1 feature utilizing MotionLayout to adapt UI elements to a given function. This could include animating control elements onto and off screen, displaying and hiding a form, or animation of complex UI transitions.  MotionLayout behaviors are defined in a MotionScene using one or more Transition nodes and ConstraintSet blocks.  Constraints are defined within the scenes and house all layout params for the animation.  The help fragment includes motion layout to enhance user experience by showing an animation on navigation to that page. |

Local and Network data

| CRITERIA | MEETS SPECIFICATIONS |
| --- | --- |
| Connect to and consume data from a remote data source such as a RESTful API. | The Application connects to at least 1 external data source using **Retrofit** or other appropriate library/component and retrieves data for use within the application.  Data retrieved from the remote source is held in local models with appropriate data types that are readily handled and manipulated within the application source. Helper libraries such as **Moshi** may be used to assist with this requirement.  The application performs work and handles network requests on the appropriate threads to avoid stalling the UI.  The application connects with The Odds API using Retrofit and data is retrieved using Moshi. A function parses the JSON array and retrieves the odds for the English Premier League football games available on the odds service. From this data, a list of real time matched bet opportunities can be produced using the different bookmakers. |
| Load network resources, such as Bitmap Images, dynamically and on-demand. | The Application loads remote resources asynchronously using an appropriate library such as **Glide** or other library/component when needed.  Images display placeholder images while being loaded and handle failed network requests gracefully.  All requests are performed asynchronously and handled on the appropriate threads.  Glide is used to retrieve the logo images of the bookmakers from remote sources. Placeholder image is displayed in the event of failure to download image. |
| Store data locally on the device for use between application sessions and/or offline use. | The application utilizes storage mechanisms that best fit the data stored to store data locally on the device. Example: SharedPreferences for user settings or an internal database for data persistence for application data. Libraries such as [**Room**](https://developer.android.com/topic/libraries/architecture/room) may be utilized to achieve this functionality.  Data stored is accessible across user sessions.  Data storage operations are performed on the appropriate threads as to not stall the UI thread.  Data is structured with appropriate data types and scope as required by application functionality.  Data is stored using an internal database utilising Room. The database serves both the storage of the users’ saved bets and also as a cache for the latest search for bets using the API service |

Android system and hardware integration

| CRITERIA | MEETS SPECIFICATIONS |
| --- | --- |
| Architect application functionality using MVVM. | Application separates responsibilities amongst classes and structures using the MVVM Pattern:   * Fragments/Activities control the Views * Models houses the data structures, * ViewModel controls business logic.   Application adheres to architecture best practices, such as the observer pattern, to prevent leaking components, such as Activity Contexts, and efficiently utilize system resources.  The application uses the MVVM pattern with a fragments, viewmodels and repository. Repository accesses database and API service. Bets are stored as models and the application utilises both a database and domain level model. |
| Implement logic to handle and respond to hardware and system events that impact the Android Lifecycle. | Beyond MVVM, the application handles system events, such as orientation changes, application switching, notifications, and similar events gracefully including, but not limited to:   * Storing and restoring state and information * Properly handling lifecycle events in regards to behavior and functionality   + Implement bundles to restore and save data * Handling interaction to and from the application via Intents * Handling Android Permissions   Databinding is used to handle the lifecycle events for the application. Data is transferred to notification builder using a bundle attached to intent. |
| Utilize system hardware to provide the user with advanced functionality and features. | Application utilizes at least 1 hardware component to provide meaningful functionality to the application as a whole. Suggestion options include:   * Camera * Location * Accelerometer * Microphone * Gesture Capture * Notifications   Permissions to access hardware features are requested at the time of use for the feature.  Behaviors are accessed only after permissions are granted.  The application uses notifications to remind the user to place bets. Asking user for permissions for notifications is not required. |