PROJECT SPECIFICATION

Political Preparedness/Custom Application

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. Answer: Using Navigation Controller, the application navigates between 3 different fragments (screens): Profile, Profile Details, and Exercises Screen The Navigation Controller is used for Fragment-based navigation and intents are utilized for Activity-based navigation. Answer: Only the navigation controller is used as there is only one activity in the application An application bundle is built to store data passed between Fragments and Activities. Answer: SharedViewModel by activityViewModels() is used in place of application bundle because a large amount of data may pass between Fragments.

CRITERIA	MEETS SPECIFICATIONS

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 effectively and efficiently across application features, avoiding nesting layouts and maintaining a flat UI structure where possible. Answer: ConstraintLayout is used to arrange UI elements. However, one of the layouts (list_item_profile.xml) contains a nested RecyclerView to display a list of data (exercises). Data collections are displayed effectively, taking advantage of visual hierarchy and arrangement to display data in an easily consumable format. Answer: ConstraintLayout is used in each layout and LinearLayoutManager is found in every Recycler View to arrange data effectively. Resources are stored appropriately using the internal `res` directory to store data in appropriate locations including `string` * values, `drawables`, `colors`, `dimensions`, and more. Answer: Done Every element within `ConstraintLayout` should include the `id` field and at least 1 vertical constraint. Answer: Done Data collections should be loaded into the application using ViewHolder pattern and appropriate View, such as `RecyclerView` . Answer: Exercise fragment, Profile fragment, Profile Details fragment all contains RecyclerView and each has their own Adapter to load data collections using ViewHolder pattern.

CRITERIA	MEETS SPECIFICATIONS

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. Answer: Displaying and hiding a Progress View from when the timer starts and ends MotionLayout behaviors are defined in a MotionScene using one or more Transition nodes and ConstraintSet blocks. Answer: In Timer Fragment, all views are constrained in fragment_timer_scene.xml's ConstraintSet blocks. Transition nodes are defined in both FragmentTimer.kt and fragment_timer_scene.xml Constraints are defined within the scenes and house all layout params for the animation. Answer: fragment_timer_scene.xml defines constraints and contains all layout params for animation for the motion layout in fragment_timer.xml

Local and Network data

CRITERIA	MEETS SPECIFICATIONS

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. Answer: Retrofit and Moshi are used to connect to an API. 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. Answer: Entities are built to match data retrieved from a remote source. The application performs work and handles network requests on the appropriate threads to avoid stalling the UI. Answer: Use withContext(Dispatchers.IO) for network request

CRITERIA	MEETS SPECIFICATIONS

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. Answer: Glide is used to load gifs coming from the remote source. Images display placeholder images while being loaded and handle failed network requests gracefully. Answer: Done All requests are performed asynchronously and handled on the appropriate threads. Answer: Done using withContext()

CRITERIA	MEETS SPECIFICATIONS

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 may be utilized to achieve this functionality. Answer: The application uses Room for data persistence. Data stored is accessible across user sessions. Answer: Data Stored can be accessible through the Repository Data storage operations are performed on the appropriate threads as to not stall the UI thread. Answer: Use Dispatchers.IO to perform Data Storage outside the main thread. Data is structured with appropriate data types and scope as required by application functionality. Answer: Done

Android system and hardware integration

CRITERIA	MEETS SPECIFICATIONS

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. Answer: One Activity with three fragments: Exercise, Profile, Profile Details. Entities for the Exercises and Templates. Retrofit and Moshi to access Remote Source. Room to locally store Exercises and Templates for data persistence. Repository to connect remote source and Room. SharedViewModel to pass data between fragments and connect fragments to the repository.

CRITERIA	MEETS SPECIFICATIONS

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 Answer: Application handles system events gracefully as data is maintain through orientation changes, app switching, and notifications. However, it does not use bundles to restore and save data because the app doesn't use intent to pass data between fragments. In addition, the only Android Permission required for the app to properly run is Internet Permission which is given.

CRITERIA	MEETS SPECIFICATIONS

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. Answer: In the Timer Fragment, a notification is sent after the timer hits 0.

Suggestions to Make Your Project Stand Out!

1. As with any mobile application, attention to detail within the UI, including animations within the screens and/or while navigating will elevate the application presentation as a whole. Proper use of visual hierarchy and consistent implementation with Styles can assist in elevating the experience. Ensuring screen real estate is properly utilized, but not overburdened will provide a positive user experience.
2. Caching data, when possible, to provide some level of application functionality when offline and/or to reduce the network burden of the application can help demonstrate and mirror real-world application goals found in many

enterprise applications at scale. As such, elevate your project by utilizing local storage and caching on network requests when it would not deter from the application experience. Providing users with choice and customization through Shared Preferences is a great way to balance real-time data vs possible performance gains by giving power to the user.

3. The mobile experience is all about personal needs and convenient access. The features of the application should reflect a personal need and provide functionality and features that reflect the solution to that need. When possible, think about the following considerations:
4. Does the application work for multiple users?
5. Does the application provide value over a website or similar static content?
6. Does the application provide a coherent user experience that effectively and intuitively guides the user's behavior?