**Architectural and Coding Guidelines for**

**ESL Android Application Using Java**

By: Team J5

Saiteja Janjirala

VamshiKishore Aemula

Nithish Sai Vardhan Gollapapani

Mahesh RamaChandrapu

Monish Bhogegowda Shylaja

Sai venkata shreyas bandaru

Supervisor:

Prof. Virginia Mushkatblat

Software Engineering Management (COMP 583)

California State University, Northridge

1. **Introduction**

The project is for adult English learners to improve their knowledge of English. It uses a contextual approach to language learning and also uses documents as entry parameters to help establish the context in which to learn. The application will be developed using JAVA programming language.

1. **Coding Guidelines**
2. Use 4 spaces per indentation level.
3. Tabs should be used solely to remain consistent with code that is already indented with tabs.
4. Use camel case for variables and functions.
5. Use Snake Case for Unit and integration tests.
6. Use enums to represent a fixed set of values instead of using constants or magic numbers or for handling states of the UI
7. Use private fields and methods to encapsulate your code and prevent external access to implementation details.
8. **Coding Language**

JAVA, XML

1. **Tools**

GitLab

Android Studio

1. **Framework/Pattern**

Android

1. **Process/ Source Control**

GitLab

1. **Operating System**

MAC OS

Windows

LINUX

1. **Policy on Conflict**

Team members are expected to maintain honest and transparent communication with both the team manager and other team members regarding the progress of assigned tasks. Team managers are responsible for managing conflicts and resolving issues in an unbiased manner. Additionally, team members are encouraged to promptly notify the team manager of any obstacles or challenges that they encounter during the project. By following these guidelines, teams can build trust, foster collaboration, and work towards successful project outcomes.

1. **Naming Conventions**

**Packages**

* Each package must specify a specific usage.
* ui: This package would contain all of the UI-related code, such as activities, fragments, and custom views.
* data: This package would contain all of the model classes, such as database models, and repository classes.
* utils: This package would contain all of the utility classes and functions, such as helper methods and custom exceptions.
* adapters: This package would contain all the adapter classes for RecyclerViews and ListViews.

**Variables, Methods, and Classes**

* Use meaningful names: Use descriptive names for variables and methods that convey their purpose and functionality.
* Use camelCase for naming variables, methods, and parameters. For example, myVariable, myMethod(), and myParameter.
* Use PascalCase: Use PascalCase for naming classes and interfaces. For example, MyClass or MyInterface.
* Prefix boolean variables with "is" or "has": This helps to differentiate them from other variables and makes the code more readable.
* Use ALL\_CAPS: Use ALL\_CAPS for naming constants. For example, MAX\_VALUE.
* Avoid abbreviations: Avoid using abbreviations unless they are well-known acronyms.

**Strings, Colors, and Icons**

* Use string resources: Store all strings in string resources to make it easier to localize the app.
* Use color resources: Store all colors in color resources to make it easier to maintain a consistent color scheme.
* Use vector icons: Use vector icons instead of bitmap images for scalability and flexibility.

**Testing**

* Write unit tests: Write unit tests for all classes and methods to ensure that the code functions as intended.
* Use test-driven development: Write tests before writing code to ensure that the code meets the intended functionality.
* Use mock objects: Use mock objects to simulate external dependencies and isolate the code being tested.

**GitLab Branch Naming Conventions**

* Use lowercase letters: Use all lowercase letters for branch names.
* Use hyphens to separate words: Use hyphens to separate words in branch names.
* Use descriptive names: Use descriptive names that convey the purpose of the branch, like contextual-list-screen or detail-screen.
* Use short names: Use short names that are easy to remember and type.

1. **Testing Frameworks**

* For writing unit tests we are using JUnit4 and for making assertions we are going to use truth library by google.

1. **CODE METRICS**

* We are assessing the code quality using the cyclomatic complexity of each function and class so to make it easier we are using codemetrics plugin available in android studio.
* Plugin : - <https://plugins.jetbrains.com/plugin/12159-codemetrics>

1. **GUI**

* **Android UI**