<Project Name>

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

[Always address Sections 2 through 6 of this template. Other sections are recommended, depending on the amount of novel architecture, the amount of expected maintenance, the skills of the development team, and the importance of other architectural concerns.]

# Architectural goals and philosophy

[Describe the philosophy of the architecture. Identify issues that will drive the philosophy, such as: Will the system be driven by complex deployment concerns, adapting to legacy systems, or performance issues? Does it need to be robust for long-term maintenance?

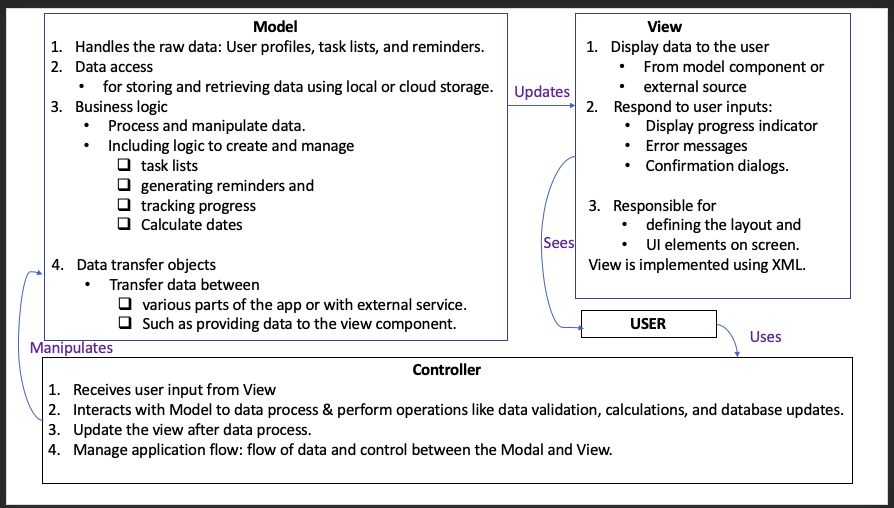
Formulate a set of goals that the architecture needs to meet in its structure and behavior. Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions?]

# Assumptions and dependencies

[List the assumptions and dependencies that drive architectural decisions. This could include sensitive or critical areas, dependencies on legacy interfaces, the skill and experience of the team, the availability of important resources, and so forth]

# Architecturally significant requirements

To realize the architecture, it is vital to achieve Separation of Concerns (SoCs) to effect scalability, testability, performance, and maintainability. Consequently, 3 major components, namely model, view, and controller, with specific responsibilities will be created.



[reference](https://www.codecademy.com/article/mvc)

# Decisions, constraints, and justifications

The decisions are based on current best practice guidelines for apps in general and for ADHD apps in particular. For instance:

1. Using a modular approach to address separation of concerns.
2. Using Kotlin to write more concise and efficient codes.
3. Thorough testing of the app.
4. Local as well as external storage capability to allow app use without internet.
5. Simple and easy navigation since the app is for individuals with ADHD.
6. [Colour coding](https://www.fullfabric.com/articles/how-to-design-visual-learning-resources-for-neurodiverse-students) to differentiate tasks or task priorities.
7. Reminders, notifications, and goal settings with flexibility to customise as needed.
8. Gamification to stimulate/maintain motivation via rewards or progress tracking.

Some constraints are:

1. Time and resources to test the app with people with ADHD.
2. Access to only free edition of third-party applications, such as Visual Paradigm.
3. Knowledge/experience/competency of team members regarding using different platforms such as Android, Kotlin and external/local database.

Justification

* The best practice approach has proven merits to positively impact the success of a project in the short run (such as debugging) as well as in the long run (scalability or creating a newer version in the future)
* Since the app is for people with ADHD, certain specific features are necessary to make the app useful; such as avoiding cluttering.
* Since certain components involve steep learning curve, it is vital to keep things simple as well as manageable to stay within allocated timeframe of the project.
* There are certain limitations to using free editions, such as Visual Paradigm, but there is no genuine need to splash money since we are university students.
* The app is designed to be Android native to allow members to focus on one platform. In addition,
  + Android has the [largest market share](https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/) worldwide.
  + Android has a well-developed ecosystem providing vital features such as frameworks and libraries.
  + Android is open-source and therefore allows greater flexibility to customise codes to suit one’s own needs.
  + Android platform provides integration with other Google products, which provides various APIs and features for app development.

# Architectural Mechanisms

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.]

## Architectural Mechanism 1

[Describe the purpose, attributes, and function of the architectural mechanism.]

## Architectural Mechanism 2

[Describe the purpose, attributes, and function of the architectural mechanism.]

# Key abstractions

[List and briefly describe the key abstractions of the system. This should be a relatively short list of the critical concepts that define the system. The key abstractions will usually translate to the initial analysis classes and important patterns.]

# Layers or architectural framework

# The ADHD Task Manager mobile application will be revolved around using a structured, multiplex architecture to ensure a compatible and viscidness design. This will allow for a clear separation of responsibilities, making it easier to create, test, and modify individual parts without altering the main functionality of the development phase. The architecture framework will include the following layers:

# 1. User Interface Layer: This will be in charge of the app's visual look and user interaction. It includes all necessary UI elements such as buttons, input fields, and menus, as well as their arrangement and visual appearance. Also, the user interface layer controls user interactions to ensure easy communication between the user and the app.

# 2. Core Functionality Layer: This will house the ADHD Task Manager app's main features. It controls tasks like creating, prioritizing, scheduling, setting reminders, and monitoring the progress of users. Additionally, the core functionality layer enables user inputs, ensuring that the information provided by users is accurate and consistent.

# 3. Data Management Layer: This will focuse on handling the app's data storage and retrieval. It communicates with the selected database or storage system to save user information, such as tasks, schedules, and preferences. The data management layer hides the underlying storage technology, creating a simple transition to an alternative storage solution if required.

# 4. External Services Layer: This area is responsible for integrating external services and APIs, like calendar synchronization, notifications, and third party productivity uses. By managing a specific layer for integration functions, the app can easily incorporate new parts without affecting the existing system.

# 5. System Infrastructure Layer: This layer controls essential sections of the application, including networking, security, and performance optimization. It ensures that the app is secure and performs efficiently under different under a variety of circumstances.

# Employing this multi-layered architecture, the ADHD Task Manager mobile app can be developed and maintained in an organized and modular manner. This approach enables an easy addition of new features, updating existing ones, and ensuring a placid user experience.

# Architectural views

To effectively portray the software architecture of the ADHD Task Manager mobile app, we will incorporate several architectural perspectives that offer a variety of views for examining and documenting architectural choices. This could possibly include:

* Logical Outlook: This viewpoint presents the structure and behavior of the system's most important features. It may cover the package organization, vital interfaces, key classes and subsystems, and the connections between these areas. If the system has a built in data persistence then the logical perspective will also display physical and logical show points of the persistent data. This perspective serves as a documented section of the overall design.
* Operational Outlook: This viewpoint demonstrates the system's physical points and the processes, threads, and components that run on those points. It offers insights into deployment and runtime areas, showing the distribution of parts across the infrastructure, their communication patterns, and potential performance or scalability problems. While this perspective may not be necessary if the system runs within a single process, it helps understand how the app works in different deployment situations and environments.
* Use Case Outlook: This viewpoint provides a list or visual aids of use cases that include architecturally significant necessities. It gives an overview of the system's functionality, focusing on points of action between users and other systems to achieve specific outputs. The use case perspective helps stakeholders understand the app's purpose and goal, and it guides the design of other architectural perspectives by putting effect on essential features.

By employing the above, we can effectively communicate the software architecture of the ADHD Task Manager mobile app to various stakeholders, making sure that the design decisions are of the same quality aligning with that of the system's requirements, objectives, and limitations. These perspectives also serve as a valuable resource for all other members of the development phase during the app's creation and maintenance sections.