ADHD Task Manager

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

The purpose of this document is to describe the design and implementation of the ADHD Task Manager, an application designed to help people with Attention Deficit Hyperactivity Disorder (ADHD) manage their daily tasks and stay focused on their goals. This document provides an overview of the philosophy, decisions, constraints, justifications, significant elements, and other overarching aspects of the system that shape its design and implementation.

The ADHD Task Manager is designed to be user-friendly and accessible, providing people with ADHD with an effective tool to manage their time and stay on track. The system includes a customizable task list, Pomodoro timer, habit tracker, rewards system, and insights and analytics. These features were carefully selected based on their ability to address the specific needs of people with ADHD.

The decisions and constraints that shaped the design of the ADHD Task Manager include the need for user-friendliness, accessibility, and effectiveness for people with ADHD. The justifications for these decisions include research on the challenges faced by people with ADHD and the effectiveness of certain features in addressing those challenges.

The significant elements of the ADHD Task Manager include the customizable task list, Pomodoro timer, habit tracker, rewards system, and insights and analytics. These elements were selected based on their ability to help people with ADHD manage their time and stay on track.

Overall, this document provides a comprehensive overview of the design and implementation of the ADHD Task Manager, highlighting the key decisions and features that make it an effective tool for people with ADHD.

# Architectural goals and philosophy

The philosophy of the ADHD Task Manager architecture is to provide a user-friendly and accessible system that effectively addresses the specific needs of people with ADHD. The architecture is designed to be robust for long-term maintenance and to adapt to changing user needs over time.

The goals of the architecture are to ensure that the system is scalable, secure, and maintainable. The system must be able to handle many users, while also providing a high level of performance and responsiveness. Security is a critical concern, and the architecture must be designed to protect user data and prevent unauthorized access. Finally, the system must be easily maintainable, with a clear separation of concerns and well-defined interfaces between components.

One critical issue that must be addressed by the architecture is the need for the system to function efficiently under unusual conditions. People with ADHD may have difficulty staying focused for extended periods, which may require the system to handle interruptions and changes to the user's schedule in a flexible and efficient manner.

Another critical issue is the need to isolate hardware dependencies from the rest of the system. The ADHD Task Manager must be designed to work on multiple operating systems, including Windows, macOS, and Linux, and must be able to adapt to changes in the underlying hardware without affecting the overall functionality of the system.

The architecture of the ADHD Task Manager is driven by the need to provide a user-friendly and accessible system that effectively addresses the specific needs of people with ADHD. The architecture is designed to be robust for long-term maintenance, scalable, secure, and maintainable, and to handle interruptions and changes to the user's schedule in a flexible and efficient manner. Critical issues such as hardware dependencies and security are addressed through clear separation of concerns and well-defined interfaces between components.

# Assumptions and dependencies

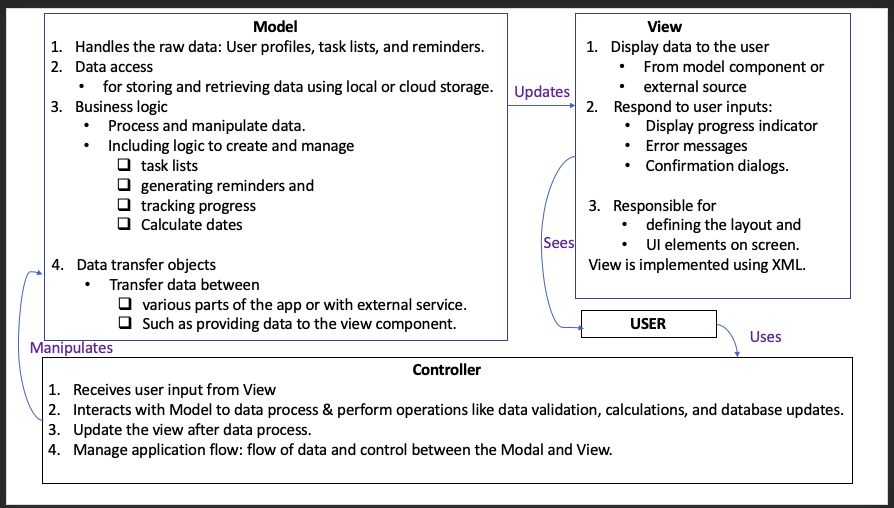
The ADHD Task Manager architecture is driven by several assumptions and dependencies that shape the design and implementation of the system. These include:

* Assumption 1: The users of the system will be using Android mobile devices.
* Assumption 2: The development team has experience in Android app development and database management.
* Assumption 3: The system will integrate with external services, such as Google Drive and Firebase, to provide additional functionality to users.
* Dependency 1: The system must be designed to work with the latest Android operating system version(s).
* Dependency 2: The system must be designed to handle many users and provide high levels of performance and responsiveness.

These assumptions and dependencies have driven the architectural decisions made during the design and implementation of the ADHD Task Manager. For example, the decision to design the app only for Android mobile devices was made to ensure a seamless and optimized user experience. The system's compatibility with the latest Android operating system version(s) was also a critical consideration during the design process. The system will also integrate with external services, such as Google Drive, Google Maps and Firebase, to provide additional functionality to users.

# 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. 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

The ADHD Task Manager will be taking a bottom-up approach when dealing with the architectural mechanisms of the project. This means that when more mechanisms are realized they will be added and implemented.

## Availability

A system to ensure that the application stays up an running over specified percentage of time. The Mechanism will ensure that including planned outages, the system will have minimal downtime.

## Archiving

This mechanism will ensure that data can be moved to storage when it reaches a specific state. In this case, it will include completed tasks as they are no longer active but are still recorded and can be viewed.

## Communication

The communication mechanism is for all elements that need to communicate with components and servers in other processes or threads. Latency, Synchronicity, size of message and protocol will need to be defined.

## Debugging

The Debugging mechanism will determine the way classes and operations are tested during creation. This is to ensure the application will be released error free. Debugging software and operational procedures will need to be defined.

## Error Management

The Error Management Mechanism is how the development team will handle errors on a user’s system. A Error reports will need to be created and sent back to the developers to analyze and fix to ensure the application works properly and so that it may not happen again.

## Graphics

This mechanism will describe how the user will see and interact with the interface. Graphics will play a key role in the success of this application, so ensuring smooth and easy to view graphical interface is very important.

## Memory Management

The Memory Management mechanism will efficiently use the device’s RAM and cache.

## Persistence

The persistence mechanism will handle writing and reading of stored data. Almost all code classes will use this mechanism.

## Security

Security mechanisms ensure that the data stored on the user’s device remains private. It will also ensure the user is safe while using the application as the application will used online.

# Key abstractions

## Task

Is the task the user will need to complete.

## Category

Is a type of task. This will be used to organize different tasks together.

## Profile

Is where information about the user will be stored and their achievements and other statistics.

## Rewards

Is what the user will receive when completing a task.

# 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.