PERSONALIZED HEALTH AND WELLNESS ASSISTANT

PACKAGE DIAGRAM, COMPONENT DIAGRAM & DEPLOYMENT DIAGRAM

Course Title – Software Engineering

Slot – G2 + TG2

Name – Vansh Kumar Payala

Registration No. – 22BAI1386

Introduction –

The ‘Personalized Health and Wellness Assistant’ will serve as a new generation mobile application that will include artificial intelligence and machine learning components to suggest a recommended health and wellness plan based on data inputted by the user. This is a new concept which will include a chatbot and will assist the users with medication and prescription guidance; therefore, the usage of the app will be enhanced while appropriate health support will be provided. The most important one is to educate and give specific recommendations concerning the state of one’s health and help to make the right decision.

To accomplish the mentioned above proposed objective of a well-structured and efficient manner of development, Software Development Life Cycle, namely the Incremental Model has been chosen. This way the application is develop in phases including this feedback of users and can be improved gradually by incorporating more features. Thus, when realizing the project in stages, it is also possible to ensure that each of the elements is functional and the final outcome is a usable application that will meet the needs of end-users. It also shows the concrete description of Incremental Model in the framework of the ‘Personalized Health and Wellness Assistant’, as well as naming all the activities of each phase and their outcomes.

Package Diagram –

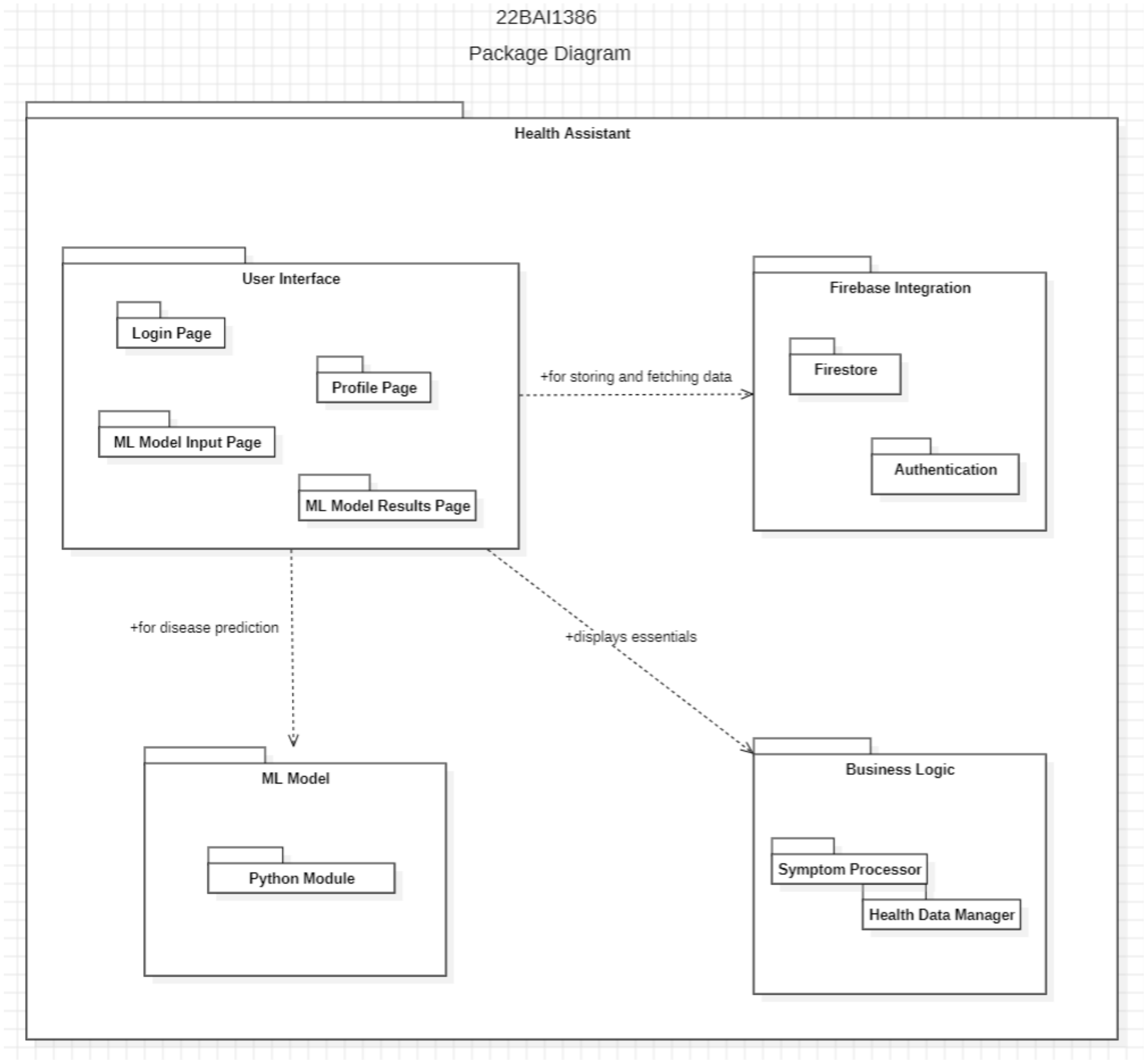
A package diagram is a type of UML (Unified Modeling Language) diagram that visually organizes the structure of a complex system by grouping related elements into packages, making it easier to understand dependencies and manage system modularity. In a package diagram, packages are depicted as large rectangles that contain other components like classes, interfaces, and other packages. Packages represent higher-level constructs, often corresponding to folders or modules in code, and serve to encapsulate related components, promoting better organization and reusability in the design. Arrows between packages indicate dependencies or relationships, showing which packages rely on or interact with each other. For instance, a package diagram in a software system might show packages for different functional layers, like "UI", "Business Logic", and "Data Access", and the dependencies between them, illustrating how changes in one package might impact others. By visualizing these relationships, package diagrams help in understanding the system architecture, maintaining code modularity, and identifying potential areas for refactoring or improvement. They are especially useful in large, complex systems where multiple developers are involved, as they provide a high-level view of the codebase's structure and dependencies.

Steps –

Packages –

- User Interface
- ML Model
- Firebase Integration
- Business Logic

Final Package Diagram –



Component Diagram –

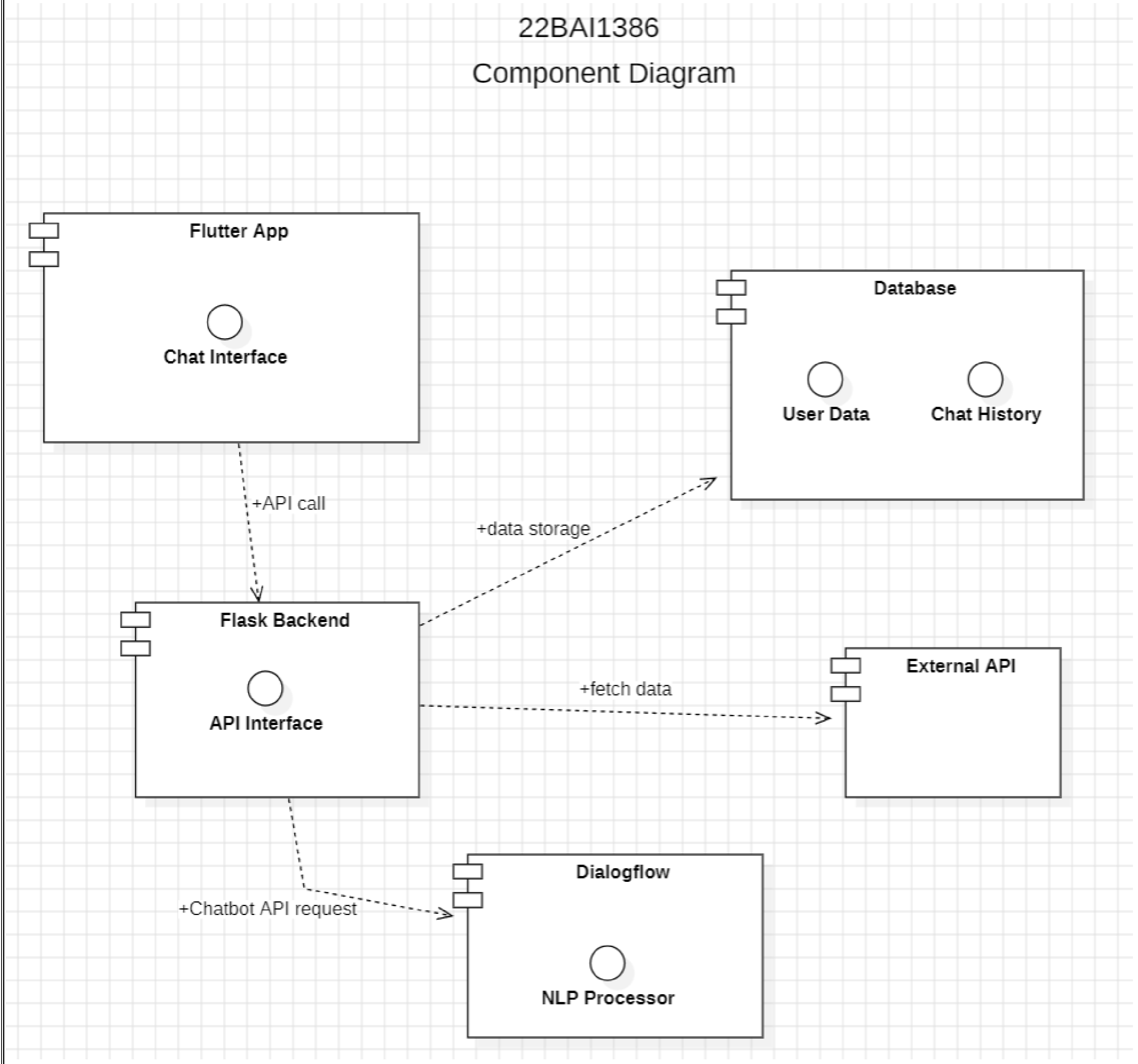
A component diagram is a type of UML (Unified Modeling Language) diagram used to depict the high-level structure and organization of a software system. It focuses on illustrating how different parts of the system, called components, interact and collaborate to achieve the system's functionality. Each component in the diagram represents a modular and replaceable unit of the system, such as libraries, modules, web services, or data sources, and has clearly defined interfaces that specify how it communicates with other components. By showing the dependencies and relationships between these components, the diagram provides insight into how data flows through the system and how different modules rely on one another. Component diagrams are especially useful for visualizing and managing complex systems, as they allow developers to see the big picture, identify dependencies, optimize for modularity, and enhance system maintainability. This makes them a key tool in the design and development of scalable, flexible software architectures.

Steps –

Components –

- Flutter App
- Flask Backend
- Dialogflow
- Database
- External API

Final Component Diagram –



Deployment Diagram –

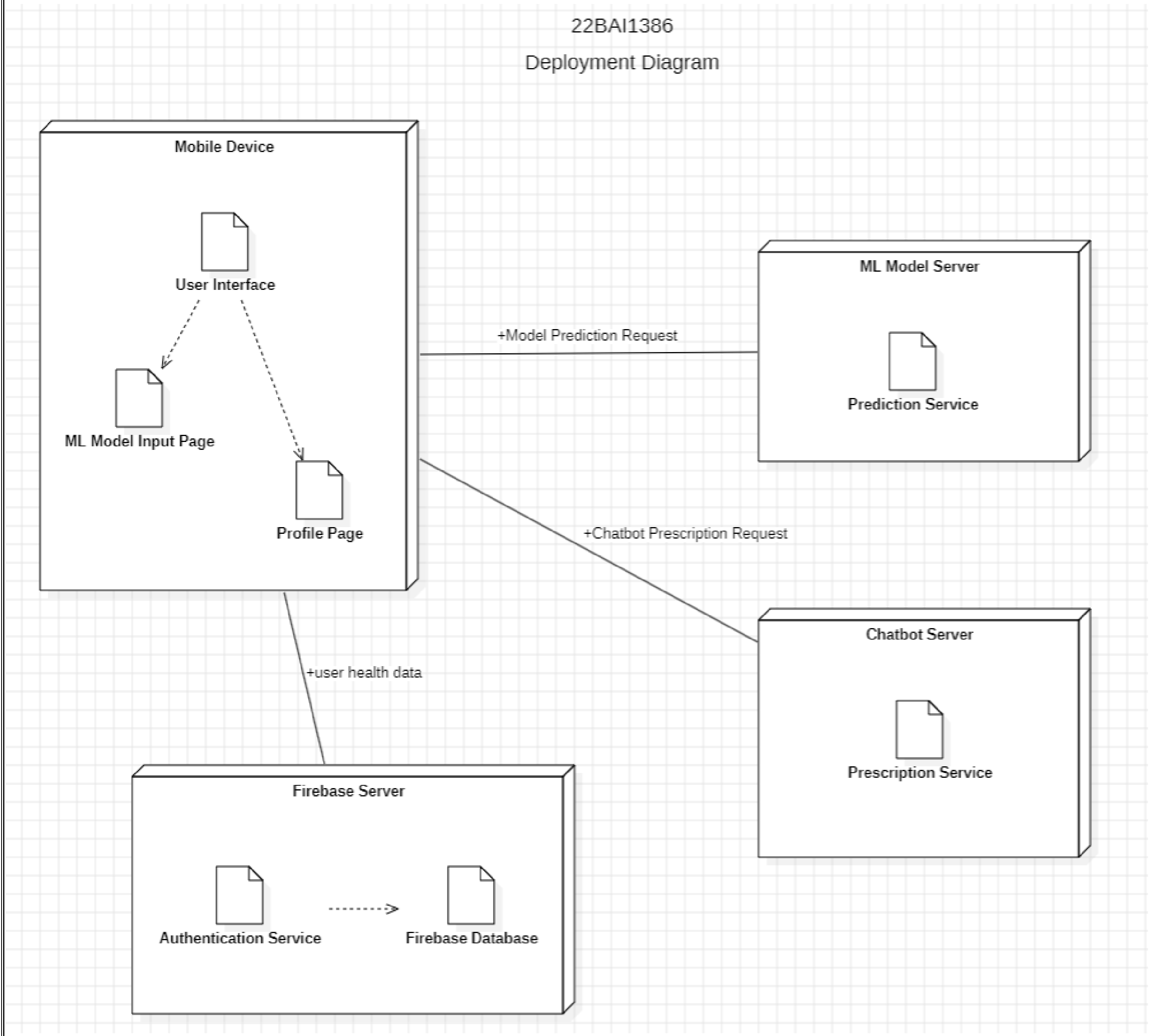
A deployment diagram is a type of UML (Unified Modeling Language) diagram used to model the physical deployment of software components on hardware nodes within a system. It provides a high-level view of the system’s hardware architecture and illustrates where each software component will reside and execute. Deployment diagrams show nodes (physical devices or servers), their connections, and the relationships between hardware and software elements, such as how applications are distributed across various machines or environments (like client devices, servers, and databases). Each node can represent real-world devices, such as web servers, application servers, or mobile devices, and may include software artifacts, like web applications or databases, deployed on them. By mapping out the physical structure, deployment diagrams are essential for understanding network configurations, resource allocation, system performance, and potential bottlenecks. They are especially valuable for planning infrastructure, optimizing resource usage, and ensuring the system meets reliability, scalability, and security requirements.

Steps –

Nodes –

- Mobile Device
- Firebase Server
- ML Model Server
- Chatbot Server

Final Deployment Diagram –


22BAI1386
Deployment Diagram
Mobile Device
User Interface
ML Model Input Page
Profile Page
ML Model Server
Prediction Service
+Model Prediction Request
+Chatbot Prescription Request
Chatbot Server
Prescription Service
+user health data
Firebase Server
Authentication Service
Firebase Database