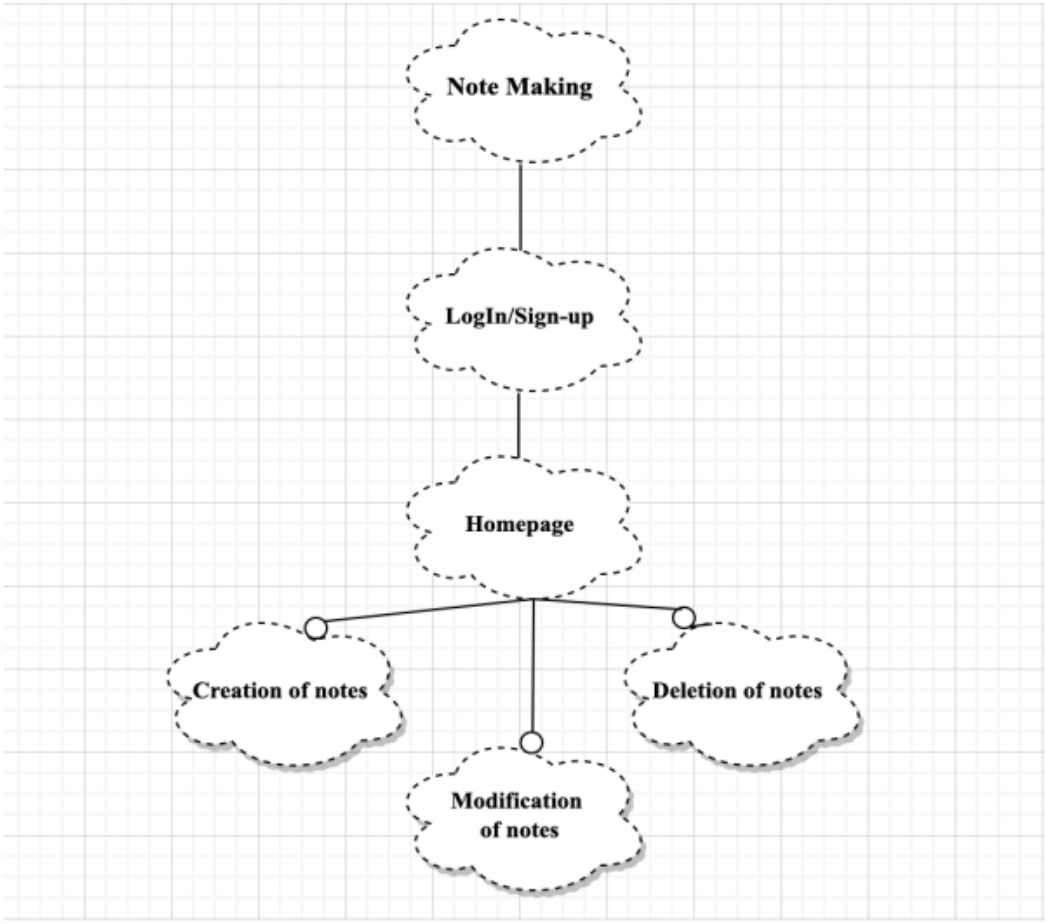
The “4+1” View Model of Software Architecture

The Logical Architecture

Description:
Functional Requirements are supported by logical architecture primarily.Examples of class diagrams for depicting logical view are Class diagrams and state diagrams.

Justification based on user requirements:
NoteBits provides the user with a platform to create and store his personalized notes under his individual login credentials. The user may be a first-time user or a frequent user, based on which a sign-up or login page is shown, respectively. These different pages are supported by different back-end technicalities.Further along in the platform, the user is equipped with a desirable interface on which he can create, store and retrieve personalized notes. The responsibility of the interface is to provide the user with appropriate options as per his needs. For example, if a user needs to create a new note, the program should interpret the same from his input and add the required information on a new blank note, and further display it for easier access later.

Diagram



The Process Architecture

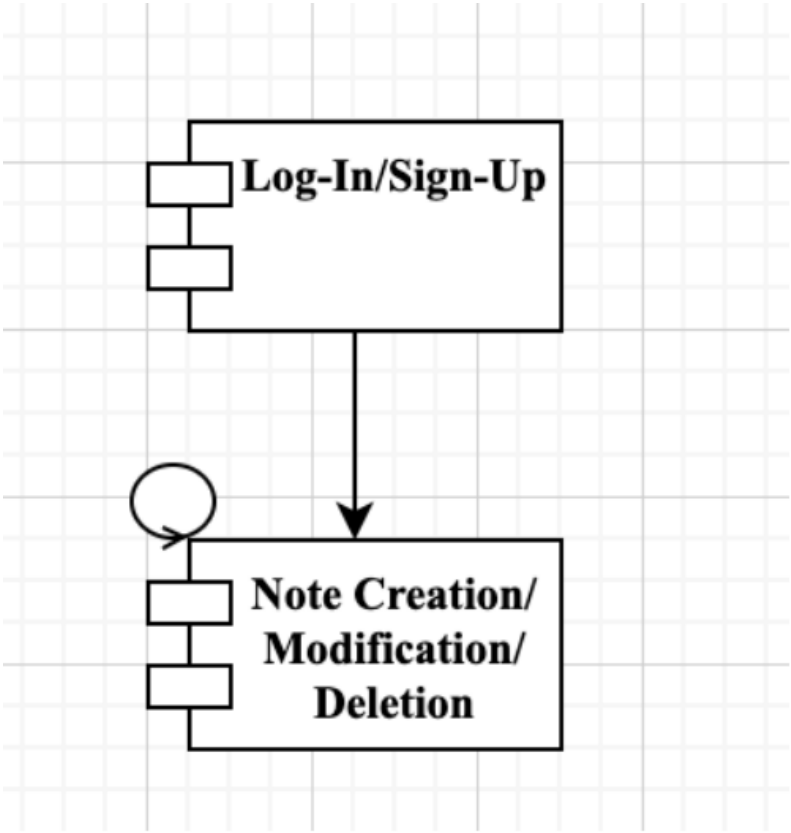
Description:

Nonfunctional requirements such as performance are taken into account in the process of architecture also availability, concurrency, distribution, and system's dynamic elements. Unified Modeling Language diagrams that can be used to describe a view process are the sequence diagram, communication diagram, and activity diagram.

Justification based on user requirements:

Considering NoteBits is a simple application providing login credentials and a platform for the user to create and store his personalized notes, the concurrency and distribution issues associated with the same are minimal. First, it needs to set a hierarchy of processes where initially a login page comes up followed by the note-making platform. Also while creating notes it must be taken into account that multiple notes are not created or some other unintended note is not modified. Also, during visualizing all notes must be displayed simultaneously. About the performance of the platform, with minimalist functionality, the latency associated with it is very small. Also, the process of login and creating a note(major functionalities) are handled in O(1) constant time and provides efficient service.

Diagram



The Development Architecture

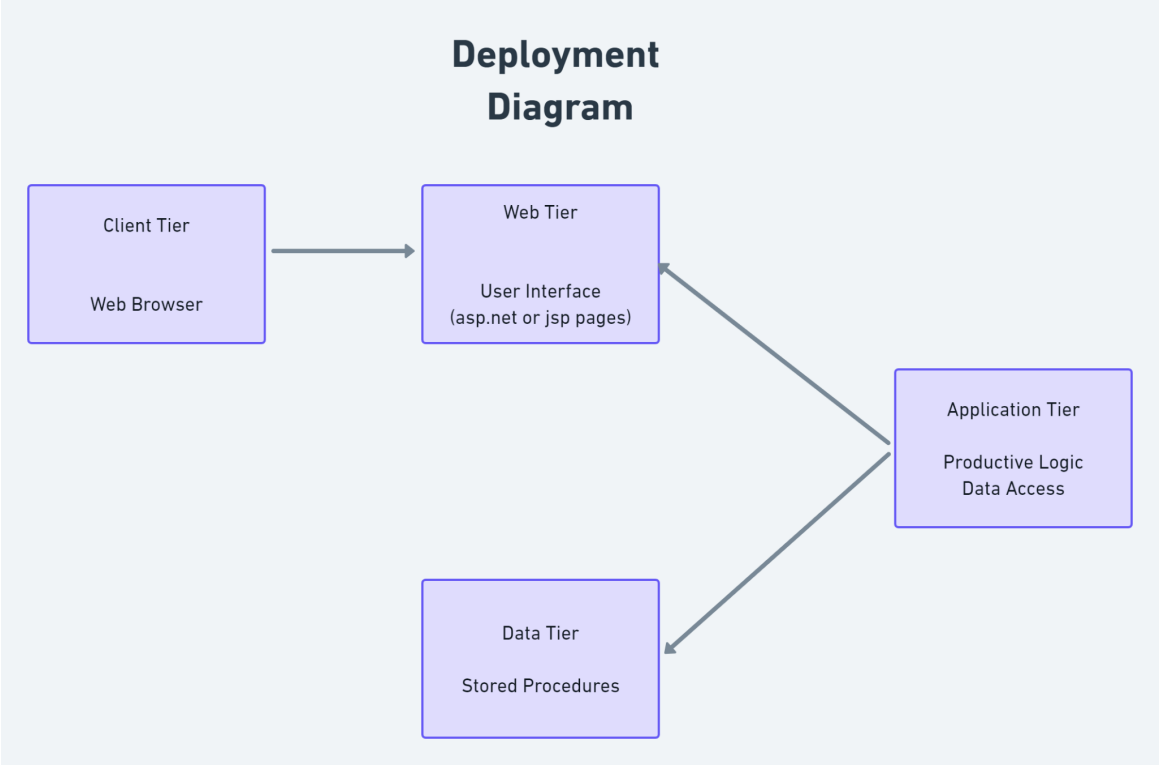
Description:

Focus is based on the genuine software module organization by the development architecture on the software development environment from the view of a programmer. For depicting the development view of UML diagrams The package diagram is among them.

Justification based on user requirements:

The program starts with a login prompt. Also, if the user is not already logged in, a sign-up option is also made available. The UI for the same is provided by HTML, CSS, and ReactJS language. The dynamics provided while typing the login information and entering to further access the program are provided by JavaScript. Once on the homepage, the UI is again provided as mentioned above. The back-end technicalities associated with clicking the button, creation of a new note, inputting information of the note, and deleting the note are provided by NodeJS. Furthermore, all the associated information is stored in a SQL database.

Diagram



The Physical Architecture

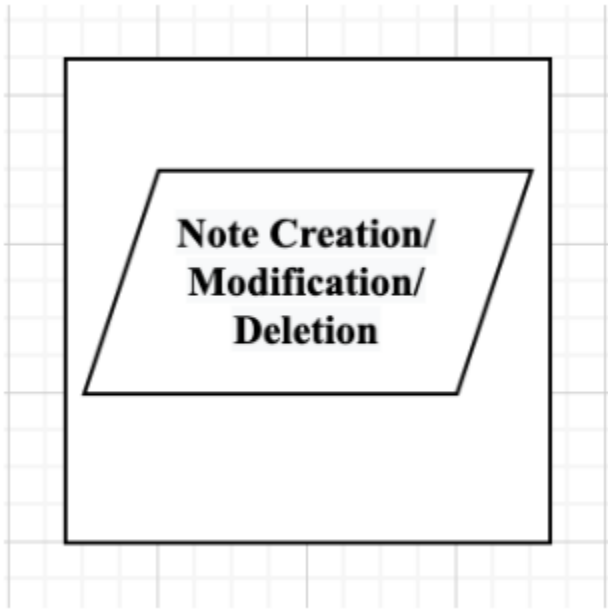
Description:

The physical configurations: some for development and testing, others for the deployment of the system for various sites are considered from the perspective of a system engineer. The

deployment diagram is one of the Unified Modeling Language diagrams for depicting physical perspective.

Justification based on user requirements:
The physical or hardware requirements of the program are linked only with storing information, login credentials, and individual notes. This information is stored in the local system of the user. Considering the deployment, the application will be open-sourced via online platforms, from where the user can access it as per his requirements and run it on his local system for regular use.

Diagram



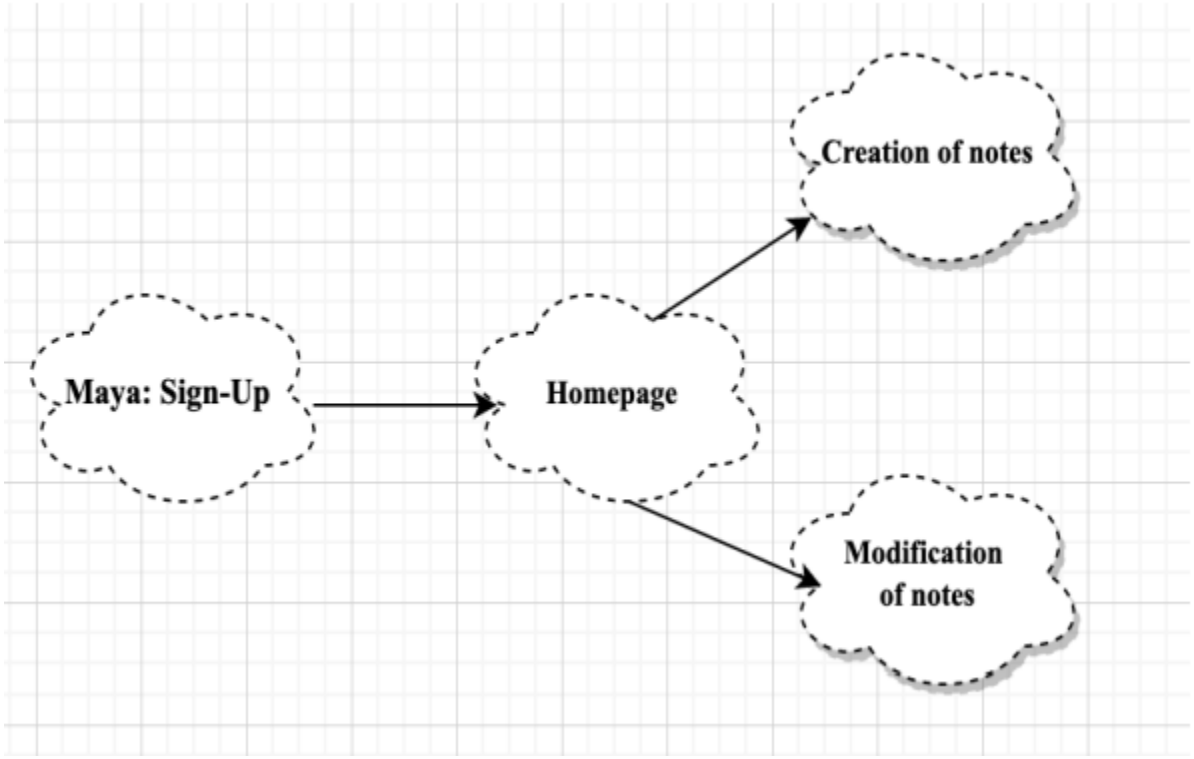
User-Scenario
Description:
The below-shown elements in the four views work together with ease, using a small set of important scenarios- examples of general use cases, used for describing corresponding scripts. Their design is expressed using object scenario diagrams and object interaction diagrams.

Justification based on user requirements:

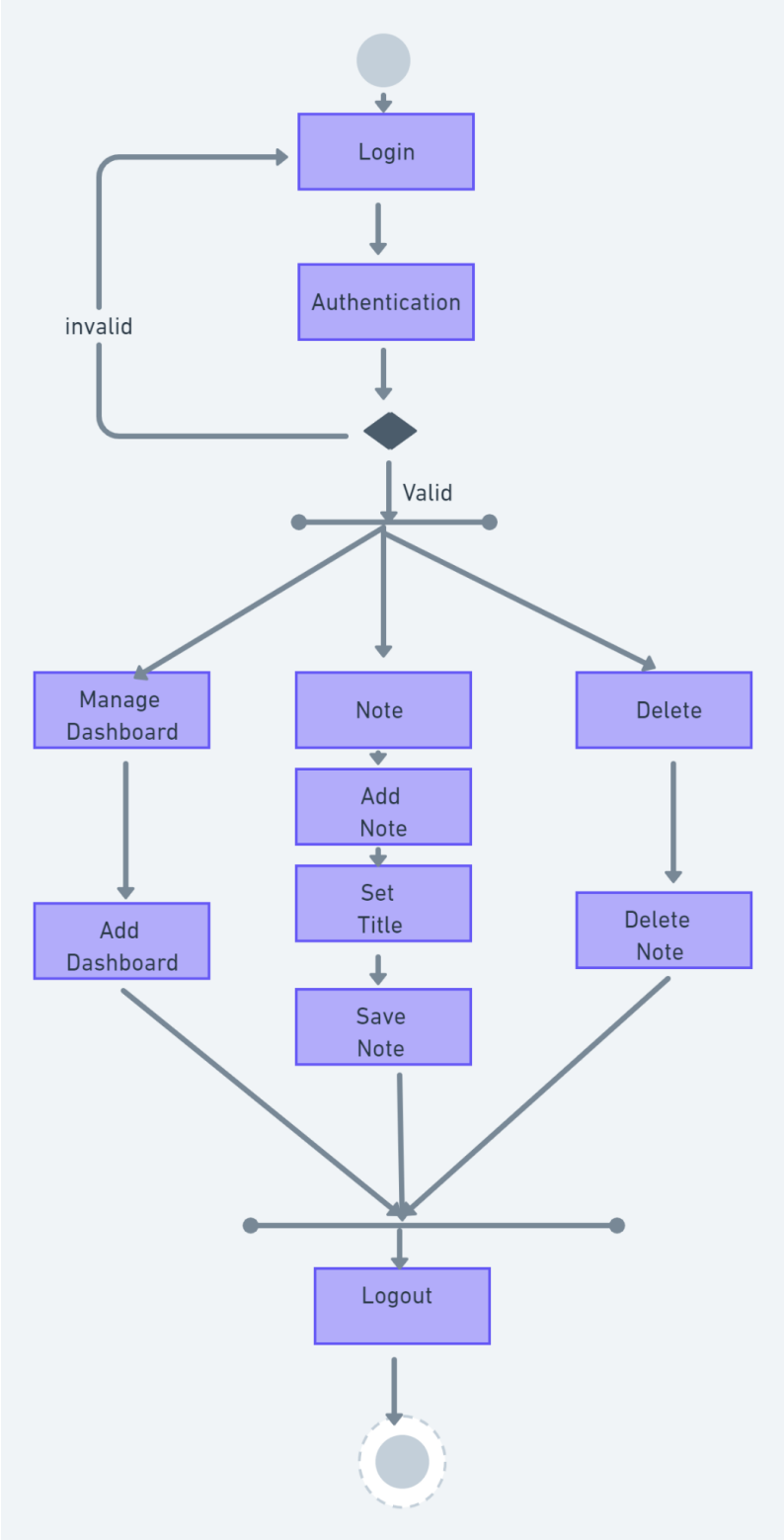
Maya is a sophomore. As a consequence, she is having a difficult time managing her classes, assignments, and quizzes. She also wants to devote a considerable amount of her time to skill development. Hence, she installs NoteBits on her system.Considering Maya is a new user while opening the program she opts for the sign-up option provided on the login page. She enters her username and password. This information is stored in her local system, a chunk of memory available to the program. She can then proceed to the homepage, where her notes-to-be will be displayed. Maya wants to make her first note and hence she clicks on the ‘plus’ button. This

Creates a new blank note. She can then edit the note providing a title and associated description. This note is then displayed on her homepage as a constant reminder and for the purpose of easier access. Maya then remembers that she wants to add some more information to the same note. She can do so as well by clicking on the note and inputting more information. Maya is satisfied for the time being and hence logs out of the program.

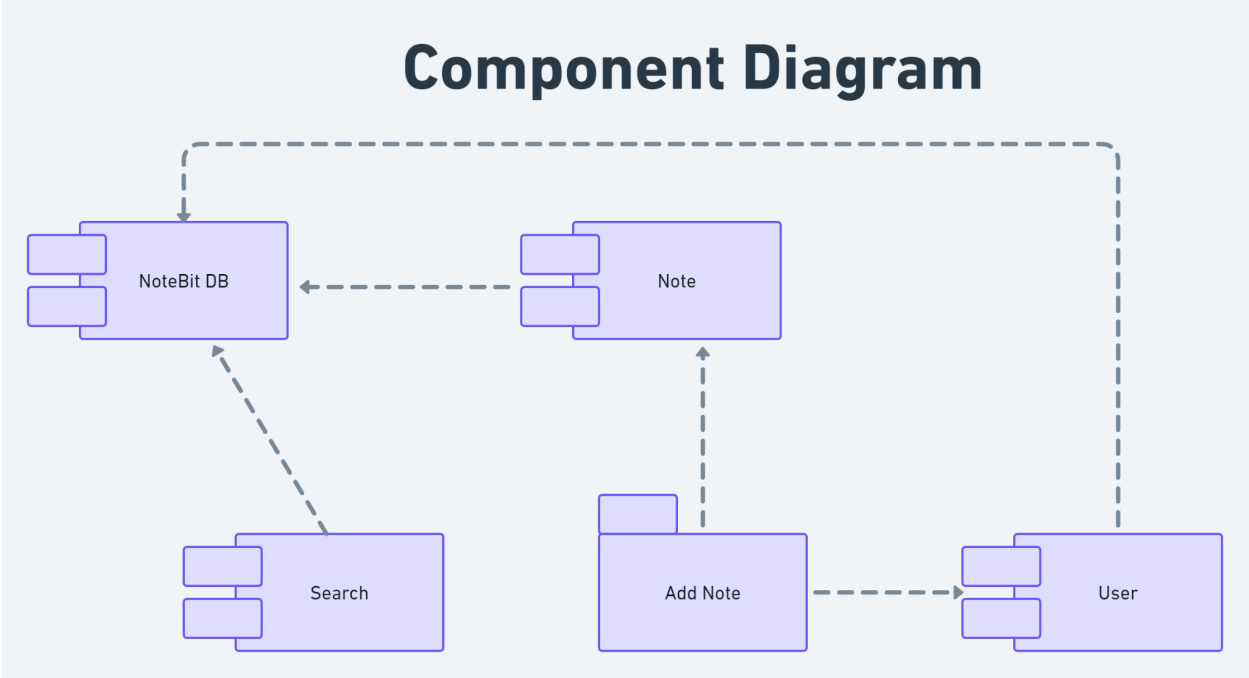
Diagram



Activity Diagram:



Component Diagram:



Sequence wise components in Sequence diagrams are: