**SETaP Coursework Iteration 2 - Template**

Team ID:

**Specification**

**For every user requirement, please fill in the template below:**

**Initial requirement**: <insert specification of user requirement as submitted back in December>

**For each system requirement associated with this user requirement, specify**:

* **Changes made to the system requirement**: <insert any changes in the specification of this system requirement, if any>
* **Rationale for the changes:** <explain why these changes were necessary>
* **Implementation progress**: <discuss the extent to which this requirement was implemented>
* **Testing progress**: <discuss the extent to which this requirement was tested>

**Design**

**Reverse engineer your code to generate an architectural model.**

The recipe management web app developed in Flutter uses a hierarchical data model with a NoSQL database. The architectural model can be reverse-engineered as follows:

**Components:**

1. User Interface (UI): The UI layer handles user input, displays recipes, and provides an interface for users to interact with the app. This layer is built using Flutter and communicates with the Business Logic Layer.
2. Business Logic Layer (BLL): The BLL contains the core logic for managing recipes, including data validation, calculations, and data manipulation. This layer is responsible for interacting with the Data Access Layer.
3. Data Access Layer (DAL): The DAL provides an abstraction layer between the Business Logic Layer and the Data Storage Layer. It handles data retrieval, storage, and updates, using the NoSQL database.
4. Data Storage Layer (DSL): The DSL is the NoSQL database that stores recipe data, user information, and other relevant data.

**Relationships between components:**

* The UI layer communicates with the BLL to perform actions such as adding, editing, or deleting recipes.
* The BLL interacts with the DAL to retrieve or update data in the database.
* The DAL communicates with the DSL to store or retrieve data.

**Data Flow:**

* User input is received through the UI layer and passed to the BLL for processing.
* The BLL performs data validation, calculations, and data manipulation, and then interacts with the DAL to update or retrieve data.
* The DAL communicates with the DSL to store or retrieve data, and then returns the data to the BLL.
* The BLL processes the data and passes it back to the UI layer for display.

**Architectural Model:**

User Interface

|

|

Business Logic

(Recipe Management)

|

|

Data Access Layer

(NoSQL Database Interaction)

|

|

Data Storage Layer

(NoSQL Database)

This architectural model shows the components, relationships, and data flow of the recipe management web app. The use of a NoSQL database provides flexibility and scalability, while the hierarchical data model and business logic layer ensure data consistency and integrity.

**Compare this model with the one submitted back in December.**

* **Are the components identified identical?**

Yes, the components identified in the current hierarchical data model are identical to those submitted in December. The same entities and data structures are being used to represent recipes, ingredients, users, and other relevant information. No new components have been added, and none have been removed.

* **Are the relationships between components identical?**

Yes, the relationships between components in the current model are identical to those in the previous model. The hierarchical structure remains the same, with recipes containing ingredients, users having saved recipes, and so on. The relationships between entities are still represented through references, maintaining the same data flow and dependencies.

* **Has the architectural pattern remained unchanged?**

Yes, the architectural pattern remains unchanged, still using a hierarchical data model. The NoSQL database continues to provide a flexible and schema-less storage solution, allowing for efficient manipulation, retrieval and storage. The overall structure and organization of the data model remain the same, with no significant changes to the architectural pattern.

* **Have any of the components been split into further sub-components?**

No, no components have been split into further sub-components. The existing entities and data structures are still used to represent the various aspects of recipe management. No new sub-entities or sub-data structures have been introduced to further break down the components.

* **Have any of the components been merged into a single component?**

No, no components have been merged into a single component. The existing entities and data structures remain separate and distinct, with no consolidation or merging of components. The same level of granularity and separation of concerns is maintained, ensuring that each component serves its specific purpose in the recipe management system.

**Implementation**

**Link to video demo:** <insert link to a 3-5 minute demo of your prototype>

**Link to GitHub repository:** <https://github.com/Tanim188/recipe\_app>

**Link to code documentation (readthedocs):** <https://github.com/Tanim188/recipe\_app/tree/main/receipe\_app\_exe>

**Link to code documentation sources (if not part of the project repository):** < https://github.com/Tanim188/recipe\_app >

**Discuss any implementation issues faced.**

**Testing**

**Link to test plan:** <https://github.com/Tanim188/recipe\_app/tree/main/test>

**Link to automated tests:** <https://github.com/Tanim188/recipe\_app/tree/main/test>

**Critical analysis**

**Provide examples of teamwork management in terms of:**

* **Leadership:** How did your team manage leadership as part of this project?, Has anything changed since the first semester?
* **Progress monitoring:** How did you keep track of the progress made as a team?, Has your approach changed from the first semester?
* **Conflict resolution:** Give examples of a couple of strategies you put in place to address team conflicts. Do these differ from the strategies used during the first semester?

**Contributions table**

Please include a contributions table specifying percentage wise each individual contribution to the work submitted. The sum of all declared percentages must be 100.