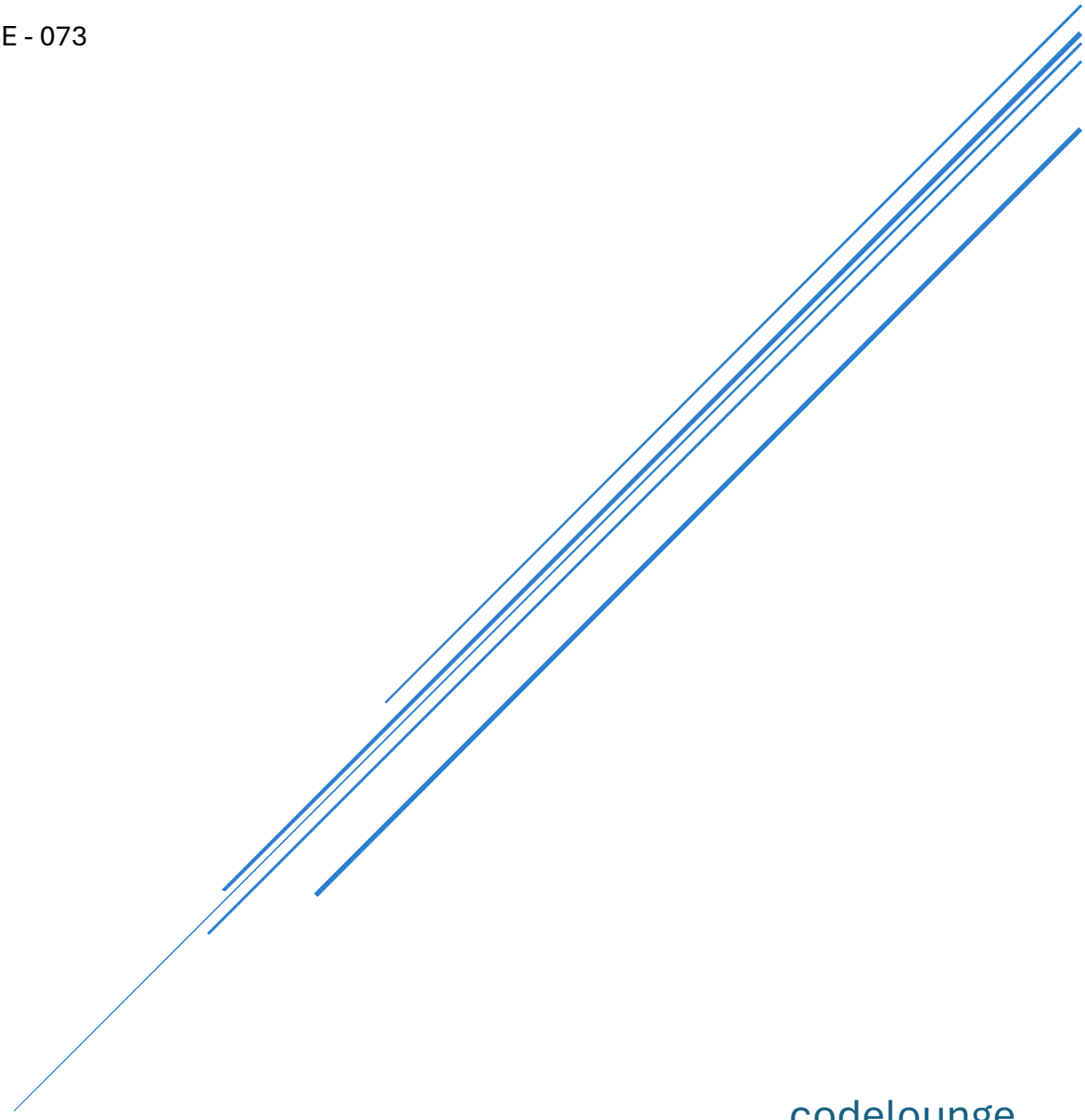
WEEK 02

Internship Report

Muhammad Mutee Ullah

FA23 – BSE - 073



codelounge

Table of Contents

Introduction:

During the second week of my internship, I focused on learning and implementing **GetX**, a powerful state management and dependency injection solution for Flutter. The main objective of this week was to understand how GetX simplifies state management, routing, and overall application architecture. This learning was essential to improve the performance, responsiveness, and scalability of the mobile application I am developing.

Duration:

Week: 2
Working Days: Monday – Friday
Daily Hours: Approximately 4–5 hours per day
Total Learning Time: Around 20–25 hours (Without Implementation)

Overview:

This week was dedicated to exploring GetX and understanding its core concepts. I started by studying:

- What GetX is and why it is used in Flutter
- Difference between traditional setState() and reactive state management
- GetX state management approaches:
    - Simple State Management (GetBuilder)
    - Reactive State Management (Obx)
- Dependency Injection using Get.put()
- Route Management using GetMaterialApp and named routes

I learned through:

- Official documentation
- YouTube tutorials
- Practical implementation in my **Todo** application

- Testing small demo examples before integrating into the main project

The primary goal was not just theoretical understanding but practical implementation.

Task Performed:

During Week 2, I performed the following tasks:

1. Installed and configured GetX in my Flutter project.
2. Replaced multiple setState() calls with GetX controllers.
3. Created separate controllers for:
    - TODO state
    - UI updates
    - Theme management
4. Implemented reactive variables using .obs.
5. Used Obx() widget for automatic UI updates.
6. Managed dependency injection using Get.put().
7. Structured the project using better separation of concerns (Controller, View, Model pattern).
8. Tested navigation using GetX routing system.
9. Optimized rebuilds to improve performance.

These tasks helped make the app cleaner, more maintainable, and performance-optimized.

Learning Outcomes:

By the end of Week 2, I achieved the following learning outcomes:

- Clear understanding of reactive state management.
- Ability to create and manage GetX controllers.
- Learned how to reduce unnecessary UI rebuilds.

- Understood dependency injection principles.
- Improved project architecture and code organization.
- Gained confidence in managing complex UI states.
- Learned how GetX improves performance and readability compared to traditional methods.

I now feel comfortable using GetX in medium to large Flutter applications.

Challenges and Solutions:

Challenge 1: Understanding Reactive Variables

Initially, I was confused about when to use .obs and how Obx() listens to changes.

Solution:
I created small practice examples to test how reactive variables update the UI automatically. After hands-on experimentation, the concept became clear.

Challenge 2: Controller Lifecycle Management

I faced issues where controllers were not being disposed properly.

Solution:
I studied GetX lifecycle methods such as onInit(), onClose(), and proper usage of dependency injection.

Challenge 3: Overusing GetX

At first, I tried converting everything to reactive unnecessarily.

Solution:
I learned best practices and used reactive state only where required, keeping the architecture balanced.

Conclusion:

Week 2 of my internship was highly productive and focused on mastering GetX for Flutter development. I successfully transitioned from basic state management techniques to a more scalable and structured architecture using GetX.

This week significantly improved my understanding of clean code practices, reactive programming, and application optimization. The knowledge gained will help me build more efficient, maintainable, and professional mobile applications in future development phases.

Note

No video is available for this week because there is no implementation, it was just learning a week.