M USMAN (SP22-BSE-073)
RE – ENGINEERING
ASSIGNMENT 4

Report on Refactoring and Migrating Google Calendar Application

1. Introduction

Google Calendar is a widely used application that allows users to schedule events, set reminders, and manage their time effectively. As technology evolves, it is crucial to refactor legacy systems and migrate them to a network-centric environment to enhance performance, scalability, and maintainability. This report outlines the steps taken to refactor the Google Calendar application, develop a migration strategy, and address potential integration challenges.

2. Code Structure Reconstruction

The legacy system of Google Calendar consists of several key components, including event management, user authentication, and notifications. To improve the code structure, the following refactoring strategies were applied:

Modularization

Large classes were broken down into smaller, more manageable modules to enhance readability and maintainability.

Design Patterns

The Model-View-Controller (MVC) design pattern was implemented to separate concerns and improve the overall architecture.

Code Cleanup

Dead code was removed, variable naming was improved, and overall readability was enhanced.

Example of Refactoring

```
# Before Refactoring
 # Before Refactoring

class EventManager:
    def __init__(self):
        self.events = []

    def add_event(self, event):
        self.events.append(event)

    def get_events(self):
        return self.events

# After Refactoring
class Event:
```

```
    def __init__(self, title, date):
        self.title = title
        self.date = date

class EventManager:
    def __init__(self):
        self.events = []

    def add_event(self, event: Event):
        self.events.append(event)

    def get_events(self):
        return self.events
```

3. Migration Strategy

Assessment
The current architecture was evaluated, and components that needed migration were identified.

Network-Centric Design
A transition to a microservices architecture was planned, allowing each service (e.g., user service, event service) to be independently deployed and scaled.

Data Migration
A strategy for migrating data from the legacy database to a cloud-based solution (e.g., Firebase, AWS DynamoDB) was developed.

API Development
RESTful APIs were created for communication between services, ensuring a smooth transition to the new architecture.

4. Integration Challenges

Data Consistency
To ensure data consistency during migration, transactions and eventual consistency models were considered.

Service Communication
Challenges in service communication, such as latency and failure handling, were addressed by implementing circuit breakers and retries.

User Experience
To maintain a seamless user experience during the transition, fallback mechanisms were provided.

5. Challenges Faced and Solutions Applied

During the refactoring and migration process, several challenges were encountered, including:

- **Challenge**: Difficulty in breaking down tightly coupled components.
  - **Solution:** Gradually refactor components while ensuring unit tests are in place to verify functionality.

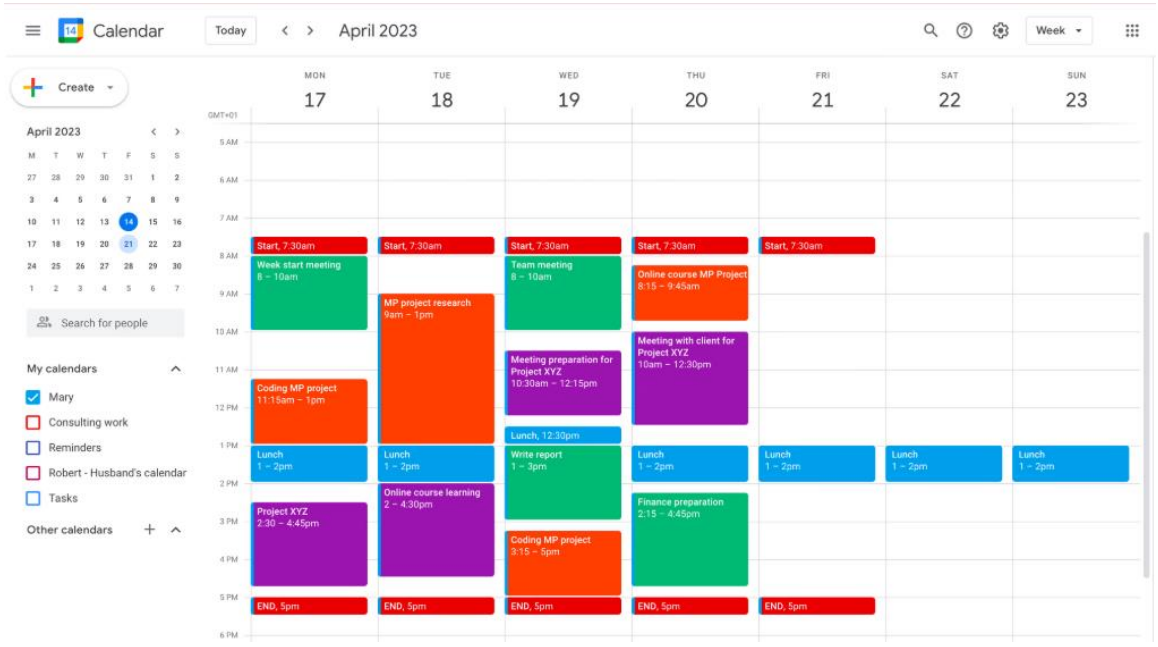
- **Challenge**: Ensuring data integrity during migration.
  - **Solution**: Implement data validation checks and use a staging environment for testing the migration process.

6. Conclusion

Refactoring and migrating the Google Calendar application to a network-centric environment is essential for improving performance and scalability. This process not only enhances the application's maintainability but also prepares it for future growth. The experience gained during this assignment has provided valuable insights into best practices for software development and migration strategies.

M USMAN (SP22-BSE-073)
RE – ENGINEERING
ASSIGNMENT 4

Screenshots



M USMAN (SP22-BSE-073)
RE – ENGINEERING
ASSIGNMENT 4