# Report 3: Core Feature Development Report (Development Phase)

## Development Progress Overview

## Implemented Features

1. Implemented Features Based on the codebase structure, the following core features appear to be implemented or are in an advanced stage of development:
   * **User Authentication**: Functionality for user login and registration, managed by LoginViewModel, RegisterViewModel, and AuthenticationService.
   * **Booking System**: Core features for searching, selecting seats, managing passenger details, and confirming bookings. This involves BookingService, BookingConfirmationViewModel, BookingDetailsViewModel, SeatSelectionViewModel, and PassengerDetailsViewModel.
   * **Customer Interface**: A dedicated user interface for customers to interact with the booking system, primarily through CustomerWindow and CustomerViewModel.
   * **Manager Interface**: A comprehensive administrative interface for managing various aspects of the train booking system, including:
     + Coach Type Management (CoachTypeManagementViewModel, CoachTypeManagementView)
     + Pricing Rule Management (PricingRuleManagementViewModel, PricingRuleManagementView)
     + Route Management (RouteManagementViewModel, RouteManagementView)
     + Seat Type Management (SeatTypeManagementViewModel, SeatTypeManagementView)
     + Station Management (StationManagementViewModel, StationManagementView)
     + Train Management (TrainManagementViewModel, TrainManagementView)
     + Train Type Management (TrainTypeManagementViewModel, TrainTypeManagementView)
     + Trip Management (TripManagementViewModel, TripManagementView) This is orchestrated by ManagerWindow, ManagerViewModel, and MainManagerViewModel.
   * **Core Data Models**: A robust set of data models representing the entities within the system, such as Booking, Coach, Passenger, Route, Seat, Station, Train, Trip, and User, located in the Models directory.
   * **Service Layer**: Dedicated service classes for various entities (e.g., BookingService, RouteService, TrainService) providing business logic and data access operations.

Technical Implementation

* **UI Framework**: Windows Presentation Foundation (WPF) is used for the graphical user interface, as evidenced by .xaml files (e.g., App.xaml, MainWindow.xaml) and the Views directory.
* **Programming Language**: C# is the primary programming language, indicated by .cs files throughout the project.
* **Data Access**: Entity Framework Core (EF Core) is utilized for object-relational mapping (ORM) and database interactions, suggested by the presence of Context.cs within the Models directory.
* **Architectural Pattern**: The project adheres to the Model-View-ViewModel (MVVM) architectural pattern, clearly separated into Models, Views, and ViewModels directories. This promotes separation of concerns and testability. BaseViewModel and RelayCommand further support this pattern.
* **Object-Oriented Principles**: The codebase demonstrates strong adherence to Object-Oriented Programming (OOP) principles, including encapsulation, inheritance (e.g., BaseViewModel), and polymorphism, through its class structure, service interfaces (e.g., IAuthenticationService), and data models.
* **Multithreading Techniques**: While not explicitly detailed in the file structure, WPF applications often leverage asynchronous programming (e.g., async/await) for UI responsiveness during long-running operations. Specific implementation details would require deeper code analysis.

## Challenges and Solutions

* Discuss major technical challenges encountered.
* Explain how the team addressed each issue.

## Git Commit History

*https://github.com/thaideptrai218/TrainBookingPRN*

## Code Quality and Documentation

* The project structure suggests an organized approach to development, with clear separation of concerns (Models, Views, ViewModels, Services). The presence of interfaces for services (e.g., IAuthenticationService) indicates a focus on maintainability and testability.
* Internal documentation and code comments would need to be reviewed directly within the source files to provide a detailed summary.

## Testing Activities

* A dedicated test project, TrainManagementWPF.Tests, exists, containing unit tests for various services (e.g., BookingServiceTests, CoachTypeServiceTests, PasswordServiceTests). This indicates that automated testing is part of the development process.
* Details regarding manual testing activities, specific test coverage, known issues, or resolution plans are beyond the scope of what can be inferred from the file structure alone.

## Next Steps

* Briefly outline planned activities for the Integration phase.