Train Ticket System

Submitted by: Alan Philpott

T00182898

Software Development - Stage 2

Date Submitted: (09/12/2016)

**Table of Contents**

[1. Introduction/overview 4](#_Toc466657121)

[2. Functional Components 5](#_Toc466657122)

[3. User Requirements 6](#_Toc466657123)

[3.1 TTSys will perform station administration. 6](#_Toc466657124)

[3.1.1 TTSys will add stations. 6](#_Toc466657125)

[3.1.2 TTSys will close stations. 6](#_Toc466657126)

[3.1.3 TTSys will list stations. 6](#_Toc466657127)

[3.2 TTSys will perform route administration. 6](#_Toc466657128)

[3.2.1 TTSys will add routes. 6](#_Toc466657129)

[3.2.2 TTSys will terminate routes. 6](#_Toc466657130)

[3.2.3 TTSys will list all routes. 6](#_Toc466657131)

[3.3 TTSys will provide scheduling options. 6](#_Toc466657132)

[3.3.1 TTSys will schedule train times. 6](#_Toc466657133)

[3.3.2 TTSys will display a timetable. 6](#_Toc466657134)

[3.4 TTSys will deal directly with tickets. 6](#_Toc466657135)

[3.4.1 TTSys will set ticket types. 6](#_Toc466657136)

[3.4.2 TTSys will perform ticket type updates. 6](#_Toc466657137)

[3.4.3 TTSys will perform ticket sales. 6](#_Toc466657138)

[3.5 TTSys will provide financial options. 6](#_Toc466657139)

[3.5.1 TTSys will provide a revenue analysis report. 6](#_Toc466657140)

[3.5.2 TTSys will provide a ticket sales analysis report. 6](#_Toc466657141)

[4. System Requirements 7](#_Toc466657142)

[4.1. System Level Use Case Diagram 7](#_Toc466657143)

[4.2. Manage Stations 8](#_Toc466657144)

[4.2.1. Add Station 8](#_Toc466657145)

[4.2.2. Close Station 9](#_Toc466657146)

[4.2.3. List Stations 10](#_Toc466657147)

[4.3. Manage Routes 11](#_Toc466657148)

[4.3.1. Add Route 11](#_Toc466657149)

[4.3.2. Terminate Route 12](#_Toc466657150)

[4.3.3. List Routes 13](#_Toc466657151)

[4.4. Train Scheduling 14](#_Toc466657152)

[4.4.1. Schedule Train 14](#_Toc466657153)

[4.4.2. Train Timetable 15](#_Toc466657154)

[4.5. Sales Processing 16](#_Toc466657155)

[4.5.1. Set Ticket Type 16](#_Toc466657156)

[4.5.2. Update Ticket Type 17](#_Toc466657157)

[4.5.3. Sell Ticket 18](#_Toc466657158)

[4.6. Generate Finance Reports 20](#_Toc466657159)

[4.6.1. Revenue Analysis 20](#_Toc466657160)

[4.6.2. Ticket Sales Analysis 21](#_Toc466657161)

[5. System Model 22](#_Toc466657162)

[5.1. Level-0 DFD 22](#_Toc466657163)

[5.2. Level-1 DFD 22](#_Toc466657164)

[5.3. Level-2 DFD (Process P1: Title) 22](#_Toc466657165)

[5.4. Level-2 DFD (Process P2: Title) 22](#_Toc466657166)

[5.5. Level-2 DFD (Process P3: Title) 22](#_Toc466657167)

[6. Data Model (Class Diagram) 23](#_Toc466657168)

[6.1. Class Diagram 23](#_Toc466657169)

[6.2. Relational Schema 23](#_Toc466657170)

[6.3. Database Schema 23](#_Toc466657171)

[7. Conclusion 24](#_Toc466657172)

[8. Appendices 25](#_Toc466657173)

[8.1. Appendix A – Title 25](#_Toc466657174)

[8.2. Appendix B – Title 25](#_Toc466657175)

# Introduction/overview

This document will outline the functional components and requirements for a Train Ticket System. The main system user will be the staff member’s at a train station. The functional activities which the staff will have control of are outlined in the Hierarchy chart. An example of one functional component is being able to add a new route to the sytem. The functional requirements are specified as both user requirements and system requirements using UML use case modelling.

The system model shows the system requirements as a collection of interacting functions and the external intities (e.g. The Customer) with which they interact. DFD’s have been used to produce the system model. The system’s data model is represented using Class Diagram’s. The class diagram is then transformed into a relational schema and then a database schema.

The primary actor for the system is the train stations staff member’s.

The system will provide station management. The primary actor can add a new station, close a station and list all stations. The system will provide route management being able to add a route, terminate a route and list a routes. A train schedule for a route can be created along with being able to view a train timetable of all scheduled trains. The system provides a function in relation to tickets. The system will firstly allow you to create/set ticket types e.g. Adult Single. If any new types should need to be added a function called Update Ticket Type will allow such updating. A ticket can then be sold in relation to the customer’s requests. The system will finally provide a function to view revenue analysis and ticket sales analysis, meaning you can view the gross amount of money being made and the what is the most frequent selling ticket type.

# Functional Components

A ***hierarchy chart*** representing the functional components of TTSSys.

# User Requirements

## TTSys will perform station administration.

### 3.1.1 TTSys will add stations.

### TTSys will close stations.

### TTSys will list stations.

## TTSys will perform route administration.

### TTSys will add routes.

### TTSys will terminate routes.

### TTSys will list all routes.

## TTSys will provide scheduling options.

### TTSys will schedule train times.

### TTSys will display a timetable.

## TTSys will deal directly with tickets.

### TTSys will set ticket types.

### TTSys will perform ticket type updates.

### TTSys will perform ticket sales.

## TTSys will provide financial options.

### TTSys will provide a revenue analysis report.

### TTSys will provide a ticket sales analysis report.

# System Requirements

Below is a diagram which shows who interacts with what part of the system. The use-case outline what happens in each process’ function.

## System Level Use Case Diagram

Use case diagram showing what each user will interact with in the system.

Manage Stations

Receptionist

Manage Routes

Customer

Process Schedules

Process Tickets

Manager

Generate Finance Reports

## Manage Stations

This module provides functions to add a station, remove a station and list stations.

### Add Station

Will provide a function to add a station.

Receptionist

Add Station

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Station** | |
| **Use Case Id** | 3.1.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Receptionist | |
| **Other Participating Actors** | None | |
| **Description** | This case will add a new station. | |
| **Preconditions** | A **Stations File** must exist for a station to be added. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The Staff Member starts the Add Station function.  **Step 3:** The staff member enters the required data:   * Station Name * Station Status * Street * Town * County * Contact Number | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be entered. * Text only fields must not contain numerical entries. * Check if each field is unique.   **Step 5:** The system assigns a station ID.  **Step 6:** The system saves the station details in the **Stations File.**  **Step 7:** Display confirmation message.  **Step 8:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A new station is added. | |
| **Post conditions** | This station may now be included on a route. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Close Station

Will provide a function to close stations.

Receptionist

Close Station

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Close Station** | |
| **Use Case Id** | 3.1.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will close a station. | |
| **Preconditions** | Stations must exist in the **Stations File** to be deleted. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The Staff Member starts the close station function.  **Step 3:** The staff member chooses the station name from a dropdown box. | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be entered. * System retrieves details of station.   **Step 5:** The system sets the station status to closed in the **Stations File.**  **Step 6:** Display confirmation message.  **Step 7:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A station is deleted. | |
| **Post conditions** | The station is no longer available for including on routes. | |
| **Business Rules** | The station cannot be closed if included on an existing route. | |
| **Implementation Constraints** |  | |

### List Stations

Will provide a function to list all stations.

Receptionist

List Stations

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Stations** | |
| **Use Case Id** | 3.1.3 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will list all stations. | |
| **Preconditions** | Stations must exist in the **Stations File**. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts the list station function.  **Step 4:** Staff member clicks finish. | **Step 2:** System retrieves details of all stations from the **Stations File** in order of station name.  **Step 3:** The system displays the UI.  **Step 5:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Staff Member Filters Data**  **Print Copy Requested** | **Step 4:** The Staff Member selects a filter keyword.   * All * Active * Closed   **Step 4:** The staff member requests a print copy. | **Step 5:** The system filters by keyword entered and displays matching station information.  **Step 5:** The system prints the list. |
| **Conclusions** | A list of all stations is displayed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Manage Routes

This module provides functions to add a route, terminate a route and list routes.

### Add Route

Will provide a function to add a route.

Receptionist

Add Route

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Route** | |
| **Use Case Id** | 3.2.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will add a new route. | |
| **Preconditions** | A **Routes File** and a **Stations File** must exist for a route to be added. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts the add route function.  **Step 3:** The staff member enters the required data:   * Set Route Status * Departure Station Code * Arrival Station Code * Total Route Distance | **Step 2:** The system retrieves all stations and displays the UI.  **Step 4:** The system validates the data entered:   * Route ID does not exist. * All required fields must be entered. * Departure station name must exist. * Arrival station name must exist. * Total Distance must be in KM. * Text only fields must not contain numerical entries.   **Step 5:** The system saves the route details in the **Routes File.**  **Step 6:** Display confirmation message.  **Step 7:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A new route is added. | |
| **Post conditions** | This route can now be used to schedule a train. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Terminate Route

Will provide a function to terminate routes.

Receptionist

Terminate Route

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Terminate Route** | |
| **Use Case Id** | 3.2.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will terminate a route. | |
| **Preconditions** | A **Routes File** must exist for an inoperative route to be terminated. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts the terminate route function.  **Step 3:** The staff member enters the required data:   * Route ID | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be entered. * Route ID must exist.   **Step 5:** The system sets the route status to terminated in the **Routes File.**  **Step 6:** Display confirmation message.  **Step 7:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A route is terminated. | |
| **Post conditions** | The terminated route is not available. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### List Routes

Will provide a function to list all routes.

Receptionist

List Routes

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **List Routes** | |
| **Use Case Id** | 3.2.3 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will list all routes. | |
| **Preconditions** | Routes must exist within the **Routes File**. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts the list route function.  **Step 4:** The staff member clicks finish. | **Step 2:** The system retrieves the details of all routes from the **Routes File** in order of Route ID.  **Step 3:** The system displays the UI.  **Step 5:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Staff Member Filters Data**  **Print Copy Requested** | **Step 4:** The Staff Member enters a filter keyword.   * All * Departure Station Code * Total Route Distance   **Step 4:** The staff member requests a print copy. | **Step 5:** The system filters by keyword entered and displays matching route information.  **Step 5:** The system prints the list. |
| **Conclusions** | A list of routes is displayed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Train Scheduling

This module provides functions to schedule a train and view a train timetable.

### Schedule Train

Will provide a function to schedule a train on a route.

Receptionist

Schedule Train

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Schedule Train** | |
| **Use Case Id** | 3.3.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will schedule a train on a route. | |
| **Preconditions** | A **Schedules File** must exist and a **Routes File** must exist. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts the schedule train function.  **Step 3:** The staff member enters the required data:   * Departure Station   **Step 6:** The staff member choses the required Route for scheduling.  **Step 7:** The staff members enters the required data:   * Train Number * Number of Carriages * Departure Time * Journey Time | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be entered. * Departure Station must exist.   **Step 5:** The system retrieves all routes for chosen departure station and displays the UI.  **Step 8:** The system adds Train Information to the **Schedules File**.  **Step 9:** Display confirmation message.  **Step 10:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A train is scheduled on a route. | |
| **Post conditions** | Tickets can be bought for a train. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Train Timetable

Will provide a function to view a train timetable.

Receptionist

Train Timetable

Customer

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Train Timetable** | |
| **Use Case Id** | 3.3.2 | |
| **Priority** | Low | |
| **Source** | Customer | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None. | |
| **Description** | This case will show a train timetable. | |
| **Preconditions** | The train must exist in the **Schedules File**. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts train timetable function.  **Step 2:** The staff member chooses the requested route.  **Step 5:** The staff member clicks finish. | **Step 3:** The system retrieves details of all train schedules with the route ID from the **Schedules File** in order of Train Number.  **Step 4:** The system displays the UI.  **Step 6:** The system resets the UI**.** |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Staff Member Filters Data**  **Print Copy Requested** | **Step 4:** The staff member enters a filter keyword.   * Train Number   **Step 4:** The customer requests a print copy. | **Step 5:** The system filters by keyword entered and displays matching Schedule Information.  **Step 5:** The systems prints the timetable information requested. |
| **Conclusions** | A train timetable is displayed. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## Sales Processing

This module will provide functions to set ticket type, update ticket type and sell tickets.

### Set Ticket Type

Will provide a function to set ticket type.

Receptionist

Set Ticket Type

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Set Ticket Type** | |
| **Use Case Id** | 3.4.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will set a ticket type for sale purposes. | |
| **Preconditions** | A **Rates File** must exist for ticket types to be set. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The manager starts the set ticket type function.  **Step 3:** The manager enters the required data:   * Ticket Type ID * Ticket Type Description * Ticket Type Rate per KM | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be entered. * Text only fields must not contain numerical entries. * Numerical value only fields must not contain text values.   **Step 5:** The ticket type data is added to the **Rates File**.  **Step 6:** Display confirmation message.  **Step 7:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A ticket type is created. | |
| **Post conditions** | A ticket can be sold in relation to a certain ticket type. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Update Ticket Type

Will provide a function to update ticket type.

Receptionist

Update Ticket Type

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Ticket Type** | |
| **Use Case Id** | 3.4.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | None | |
| **Description** | This case will update data for a ticket type. | |
| **Preconditions** | The **Rates File** must contain entries for data to be updated. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The manager starts the update ticket type function.  **Step 3:** The manager enters the required data:   * Ticket Type ID   **Step 6:** The manager enters the new required data:   * Ticket Type Name * Ticket Type Description * Ticket Type Rate per KM | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be entered. * Ticket Type ID must exist.   **Step 5:** The system displays next UI.  **Step 7:** The system validates the data entered:   * All required fields must be entered. * Text only fields must not contain numeric entries.   **Step 8:** The ticket type is updated.  **Step 9:** Display confirmation message.  **Step 10:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected**  **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3.  **Step 7:** Invalid data detected.  **Step 8:** Display an appropriate error message and return to step 6. |
| **Conclusions** | A ticket type is updated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Sell Ticket

Will provide a function to sell tickets.

Receptionist

Sell Ticket

Customer

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Activity Diagram - Sell Ticket.png

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Sell Ticket** | |
| **Use Case Id** | 3.4.3 | |
| **Priority** | High | |
| **Source** | Customer | |
| **Primary Business Actor** | Staff Member | |
| **Other Participating Actors** | Customer | |
| **Description** | This case will sell and print a ticket and record the sale in a **Sales File**. | |
| **Preconditions** | A **Schedules File**, **Rates File** and **Sales File** must exist for tickets to be sold. Valid student card identification must be presented. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The staff member starts the sell ticket function.  **Step 4:** The staff member selects the appropriate data relevant to the customer:   * Departure Station   **Step 7:** The staff member selects the appropriate route relevant to the customer.  **Step 10:** The staff member selects the appropriate ticket type relevant to the customer.  **Step 13:** The staff member confirms purchase. | **Step 2:** The system retrieves all stations in order of name.  **Step 3:** The system displays the UI.  **Step 5:** The system retrieves all routes from the **Routes File** with the relevant departure station name from the **Stations File**.  **Step 6:** The system displays the UI.  **Step 8:** Retrieve ticket types from **Rates File**.  **Step 9:** The system displays the UI.  **Step 11:** The system calculates the cost of the ticket.  **Step 12:** The system displays confirmation of ticket information the cost of the ticket.  **Step 14:** The sale is recorded in the **Sales File.**  **Step 15:** The system prints the ticket**.**  **Step 16:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Customer Declines Ticket** | **Step 13:** The customer declines the ticket.  **Step 15:** The receptionist confirms cancellation. | **Step 14:** The system displays a warning message.  **Step 16:** Display an appropriate confirmation message and return to step 4. |
| **Conclusions** | A ticket is sold. | |
| **Post conditions** | This ticket can now be used on a train. | |
| **Business Rules** | ID verification must be shown on request to certain ticket types. | |

## Generate Finance Reports

This module provides functions to view revenue analysis and ticket sales analysis.

### Revenue Analysis

Will provide a function to view revenue analysis.

Manager

Revenue Analysis

Validation

<< Includes >>

Display Error Message

<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Revenue Analysis** | |
| **Use Case Id** | 3.5.1 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This case will provide a revenue analysis report. | |
| **Preconditions** | A **Sales File** must exist. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The manager starts the revenue analysis function.  **Step 3:** The manger selects the required option:   * Specific Date * Time Period | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be selected/entered. * Date cannot be in the future.   **Step 5:** The system displays the chosen UI.  **Step 5:** The system displays data for the requested information**.**   * Gross Amount   **Step 6:** Display confirmation message.  **Step 7:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A revenue analysis report is generated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### Ticket Sales Analysis

Will provide a function to view ticket sales analysis.

Manager

Ticket Sales Analysis

Validation

<< Includes >>

Display Error Message

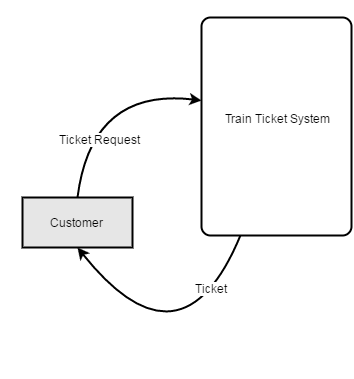
<< Extends >>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Ticket Sales Analysis** | |
| **Use Case Id** | 3.5.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | None | |
| **Description** | This case will provide a ticket sales analysis report. | |
| **Preconditions** | A **Sales File** must exist. | |
| **Trigger** | No. | |
| **Expected Scenario** | **Staff Member** | **System Response** |
|  | **Step 1:** The manager starts the ticket sales analysis function.  **Step 3:** The manager selects the required information:   * Specific Date * Time Period | **Step 2:** The system displays the UI.  **Step 4:** The system validates the data entered:   * All required fields must be selected/entered. * Date cannot be in the future.   **Step 5:** The system displays data for the requested information**.**   * Number of Adult Tickets sold. * Number of Student Tickets sold. * Number of Child Tickets sold.   **Step 6:** Display confirmation message.  **Step 7:** The system resets the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Detected** |  | **Step 4:** Invalid data detected.  **Step 5:** Display an appropriate error message and return to step 3. |
| **Conclusions** | A ticket sales analysis report is generated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# System Model

*The following dataflow diagrams have been produced for the system*:

## Level-0 DFD



## Level-1 DFD

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\DFD Level 1.png

## Level-2 DFD (Process P1: Manage Stations)

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\DFD Level 2 - Process 1.png

## Level-2 DFD (Process P2: Manage Routes)

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\DFD Level 2 - Process 2.png

## Level-2 DFD (Process P3: Train Scheduling)

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\DFD Level 2 - Process 3.png

## Level-2 DFD (Process P4: Sales Processing)

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\DFD Level 2 - Process 4.png

## Level-2 DFD (Process P5: Generate Finance Reports)

C:\Users\Alan\AppData\Local\Microsoft\Windows\INetCacheContent.Word\DFD Level 2 - Process 5.png

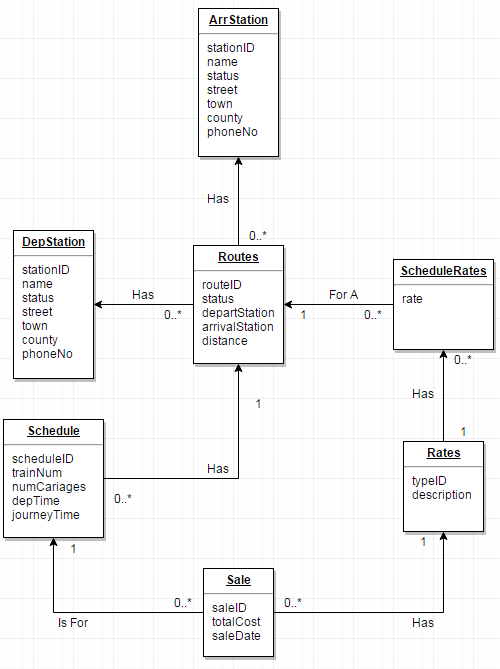
# Data Model (Class Diagram)

The following data model has bee produced for the system. Below is also the relational schema and database schema for the system.

## Class Diagram

Object Model – UML Class Diagram

Class diagram shows objects & attributes



## Relational Schema

Relational schema for the data requirements - Using ***bracket notation***

Station (stationID, name, status, street, town, county, phoneNo)

Route (routeID, depStation, arrStation, status, distance)

Schedule (scheduleID, *routeID*, trainNum, numCarriages, depTime, journeyTime)

ScheduleRates (routeID, typeID, rate)

Rates (typeID, description)

Sale (saleID, *routeID*, *typeID*, totalCost, saleDate)

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key and other constraints to be implemented.

**Schema Name:** TrainTicketSystem

**Relation:** Station

Attributes:

stationID number(5)

name char(20) NOT NULL

status char(1) NOT NULL DEFAULT ‘A’

street char(20) NOT NULL

town char(20) NOT NULL

county char(20) NOT NULL

phoneNo char(20) NOT NULL

**Primary Key:** stationID

**Relation:** Route

Attributes:

routeID number(5)

departStation char(20) NOT NULL

arrivalStation char(20) NOT NULL

status char(1) NOT NULL DEFAULT ‘A’

distance number(10,2)

**Primary Key:** routeID

**Relation:** Schedule

Attributes:

scheduleID number(5)

routeID number(5)

trainNumber number(10)

numCarriages number(2)

depTime time

journeyTime time

**Primary Key:** scheduleID

**Foreign Key:** routeID REFERENCES Route

**Relation:** Rates

Attributes:

typeID number(5)

description char(20)

**Primary Key:** typeID

**Relation:** ScheduleRates

Attributes:

routeID number(5)

typeID number(5)

rate number(5,2)

**Relation:** Sale

saleID number(5),

routeID number(5),

typeID number(5),

totalCost number(5,2)

saleDate date

**Primary Key:** saleID

**Foreign Key:** routeID REFERENCES Route

**Foreign Key:** typeID REFERENCES Rates

# Conclusion

The Train Ticket system is a wide spread system to be used at every train station to manage Stations, Routes, Schedules and Tickets.

A shortcoming of the system would be the fact that a route can only be created between 2 stations. Future development might allow for intermediate stations on routes to be added. Another shortcoming would be that only one type of ticket can be sold at once. Future development might allow for multiple types of tickets to be sold at once.

# Appendices

## Appendix A – Title

## Appendix B – Title

Might include:

* **Lookup / Reference tables**
* **Sample reports / Listings**