Requirements Analysis Document:
LogBook

JAMES Project
15-413 Software Engineering
Fall 1997
Carnegie Mellon University
Pittsburgh, PA 15213

Revision History:
Version 1.1.1 - Last Updated Nov. 19, 1997 by Michael Poole

Preface:
This document addresses the requirements of the JAMES system. The intended audience for this document are the designers and the clients of the project.

Target Audience:
Client, Developers

JAMES Members:
Gordon Cheng, Li-Lun Cheng, Christopher Chiappa, Arjun Cholkar, Uhyon Chung, Aveek Datta, John Doe, Phillip Ezolt, Eric Farng, William Ferry, Sang Won Ham, Kevin Hamlen, Pradip Hari, Yenni Kwek, Tze Bin Loh, Alexander Lozupone, Christopher Lumb, Vincent Mak, Darren Mauro,
1.0 General Goals

Business travelers frequently need to keep good records of their travels, either in their own cars or in those they rent, in order to report the costs to their companies or to the IRS. Rental car companies also have an interest in such monitoring, as do people keeping a tight household budget. To serve the needs of these diverse groups of people, the JAMES LogBook Assistant will allow detailed and accurate recording of trips taken in Mercedes vehicles. Understanding that different drivers have different travel-monitoring needs, it will be flexible enough to distinguish between business and pleasurable travel, to accommodate personal or rental cars, and to provide for a level of information security appropriate to its use. Above all, it will always be safe, reliable, and easy to use.

2.0 Current System

There is no current system. However, there are existing systems that provide functionalities that are similar to this system, albeit not on the car. For example, there is a software package called Success+ from JEANius software that allows one to keep track of car trips by logging the date and the odometer readings. You could also make annotations of each trip and whether it was a personal or a business trip. The trip logging module is a part of a bigger system that manages a marketing business.

3.0 Proposed System

3.1 Overview

The LogBook Assistant will maintain a record of trips made using a car, including (at least) the following information for each trip:

- Starting point
- Destination point
- Time elapsed
- Distance traveled
- Gas used
- Average and maximum speed

It will also allow this information to be formatted and printed for use in filling out corporate expense vouchers or tax forms. To facilitate management of a fleet of cars, it will allow for two levels of access privileges: one for a driver (who may view the record of his or her own trips) and one for an administrator (who may view all recorded trip information and modify it as appropriate).

The LogBook Assistant will require Authorization, Odometer reading, and GPS services. Currently, it will not provide any services to other components.

3.2 Functional Requirements

- Record information about vehicle usage, including origin & destination of trips, gas usage, vehicle identification, driver identification, date, time, and distance traveled. Limit access to this information based upon the user's identity as either a driver or administrator of the vehicle.
- Store all information on the car's onboard systems and allow it to be transferred to other platforms using the SmartCard.
- Allow users to retrieve the LogBook information.
- Allow the driver to start, end, suspend, and resume trips.
- Allow the user to use external systems to generate standard forms, such as tax forms, based on LogBook information.
- Distinguish between business and private trips.
- Allow the user to annotate trips.
- Allow multiple drivers to record trips on the same vehicle.
- Allow a driver to record trips on multiple vehicles.

### 3.3 Nonfunctional Requirements

#### 3.3.1 User Interface and Human Factors

The LogBook will mainly be used by two types of operators: users and administrator which may or may not be the same person. The user interface (UI) should be easy to learn but should provide any appropriate shortcuts. The component on the external system has administrative capabilities and should warn the administrator if they do anything that would provide a similar warning on an office app.

The IO devices for the LogBook system are in two places. The system in the car will have a touch screen. On the touch screen, buttons and other UI elements need to be large enough to be able to use them without the fine granularity of a stylus or a mouse. Speech will function as a secondary UI system. Depending on the speech system, this may or may not be foolproof; further analysis requires knowledge of the particular speech system. Eitherway, the speech system should be seen as an augmentation to the UI; there shouldn't be any part of the interface that requires speech and others that require use of the touch screen.

The user interface to the application that runs on the external system will look like a typical application and use traditional UI elements like menu bars, scroll bars and the like. Its user interface will have a look and feel akin to most other office applications such as Word and Quicken. In both locations, the UI will be integrated with the interfaces of the other cardlets.

#### 3.3.2 Documentation

The following documentation pieces are to be delivered as specified in the JAMES System Problem Statement.

<table>
<thead>
<tr>
<th>Document</th>
<th>Target Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Project Management Plan</td>
<td>Designers, client</td>
</tr>
<tr>
<td>Requirements Analysis Document</td>
<td>Project management</td>
</tr>
<tr>
<td>System Design Document</td>
<td>Designers</td>
</tr>
<tr>
<td>Object Design Document</td>
<td>Designers</td>
</tr>
<tr>
<td>Text Manual</td>
<td>Designers, client, maintainers</td>
</tr>
<tr>
<td>User Guide</td>
<td>Client</td>
</tr>
</tbody>
</table>

A designer is a member of the JAMES project (i.e.: the LogBook team members), the client is Daimler Benz, and the maintainers are those who may work on this project in the future.

#### 3.3.3 Hardware Consideration

The LogBook application will have pieces running on three different hardware platforms: the car's computer, the SmartCard, and the user's computer. The logging of trips will be done on the car's on-board computer, and the tax forms will be printed on the user's computer or prepared by a service center. The only thing left for the SmartCard to do is transfer trips between the user's computer and the car.

The car's computer is comparable to a desktop personal computer, probably with a real-time operating system, and should be capable of running full Java. Also, the storage capacity should be much greater than what the SmartCard has, however space will still be a consideration in this application.
The SmartCard itself runs a watered down version of Java with no threads, no garbage collection, no exceptions, or much of anything that makes Java useful. The card has a total of 2.8KB of space available for our program and any data we need to store. The other risk with the card is that the Java operand stack only has 32 bytes. So, you can't have a deep call tree, use class hierarchy, or have lots of local variables. The code that goes on the card has to be without excess. Making a good, reusable, application to run on the card will not be easy.

The user's computer, which handles the tax forms and final storage of trips, should present many design challenges. Talking to the card maybe somewhat difficult, as it is necessary to go through an old DLL, but any ugliness there can be encapsulated. Once the communication is taken care of the tax form generator shouldn't in any way stress the resources of a modern desktop computer.

Note the hardware information in here can be found in the CyberFlex lecture notes available at http://cascade1.se.cs.cmu.edu/JAMES/J_lectureFiles/cyberflex/sld001.htm.

3.3.4 Performance Characteristics

There are are no significant constraints on the storage space of the vehicle. The reason is because we are given a relatively large storage device on the vehicle. However, there is the minor possibility of the storage space on the vehicle overflowing through heavy use of the vehicle. Therefore, this case needs to be taken in to consideration.

The storage on the card is very limited, therefore the amount of data per trip to be stored on the card will need to be limited. A type of a compression along with a carefully designed data structure needs to be implemented.

The application on the vehicle needs to have a relatively fast response time less than a second to a users' input. However, this does not require much effort because the main operations of the LogBook systems will mainly be load and store operations.

3.3.5 Error Handling and Extreme Conditions

3.3.5.1 Error Handling

There are three components to the LogBook Assistant software: the on-board computer code, the personal computer code, and SmartCard transfer code. These are subject to various types of input errors, to be discussed separately. The onboard computer code communicates with the user (during driving) via three buttons: "Start Trip," "Pause Trip" and "End Trip." In each state (in a trip, or not in a trip) the behavior of these is well-defined: pressing "Start Trip" when a trip is in progress shall start a new trip, pressing "Pause Trip" when a trip is in progress shall suspend or resume the trip (as appropriate), and do nothing if no trip is in progress, and pressing "End Trip" when no trip is in progress shall be ignored. The onboard computer code communicates with the administrator (or the server, when retrieving trip information) via the onboard touch screen. The data retrieval mode (copying information onto the card) will be designed to eliminate, as much as possible, any opportunities to make invalid input: trips or legs which may be retrieved or deleted are displayed in a list box, and no buttons may be pressed for invalid operations (e.g., if the user does not have permission to delete a trip's information, the 'delete' button shall be either disabled or hidden). The personal computer component of the system has the most complicated user interface, since it must allow the user to select trips for which to print tax deduction forms and possibly to edit trip data (join or split trips). Until the functionality of this component is determined, input errors and their handling cannot be fully analyzed.

3.3.5.2 Exceptional Conditions

'Exceptional conditions' in the context of the LogBook assistant are assumed to be limited to starting or stopping a trip very shortly after starting the car or very shortly before stopping the car (i.e., turning the motor on or off). In these cases, the trip's data is 'rounded off' to reflect the car's location when the engine starts or stops, rather than when the 'Start Trip,' 'Pause Trip' or 'End Trip' buttons are pressed. The conditions for performing such round-off are not yet defined, but a suggested possibility is that the odometer has not changed by more than 0.1 miles. It has also been suggested that this be possible to change after manufacture -- perhaps by taking the car into a service center to change the onboard computer settings.

3.3.6 System Interfacing

3.3.6.1 Input
The LogBook will be notified by the Vehicle Subsystem when the car is started or shut down. It will request user authorization from the Authorization Subsystem. It will interact with the user via the user interface specified by the HCI subsystem, on both the car and external system. In addition, the LogBook will have the ability to read from the odometer, fuel gauge, clock, GPS system, and possibly other components managed by the Vehicle Subsystem.

3.3.6.2 Output

When the LogBook is not functioning properly, the user will be notified, and if a Diagnostics subsystem existed, it would be notified. However, such a subsystem is outside the scope of this project.

3.3.6.3 I/O Format and Medium

Specifying the interfaces with other JAMES components (Authorization, Travel, etc.) and with car systems (odometer, fuel system, etc.) must follow specifications governing the JAMES system architecture. The interface with user must follow specifications governing HCI (human-computer interaction) for the JAMES system. Forms will be printed through standard Java printing interface.

3.3.7 Quality Issues

The LogBook system needs to be extremely reliable because it is the intention of Daimler-Benz to install the system in many vehicles. As such, a simple error in the LogBook system would be extremely expensive to correct. Further, companies will rely on the data that the LogBook system provides to generate tax forms and so the system should reliably provide correct trip information.

The system must trap faults because of this high reliability requirement and because of its tight integration with the other systems. For example, when the LogBook system queries the odometer and a fault is generated, the LogBook system cannot simply ignore it. Rather, it must do something intelligent such as query again, or indicate that the current trip might not be accurate.

The high reliability requirement suggests that the system must be restarted as soon as possible after a failure. If the system goes down, it may miss a request to end or start a trip from the user of the system. The maximum acceptable amount of time after a failure is impossible to quantify - it is simply equal to the minimum amount of time after the failure and before a significant event occurs (user presses a start or end trip button, car starts or shuts down, etc.) Since this amount of time is impossible to establish the system needs to be restarted immediately after a failure.

3.3.8 System Modifications

There are a number of parts of the Logbook system that are good candidates for later modification.

In the user interface subsystem the current design calls for a number of 'Trip' related buttons, as more research is done in this system, the interface may evolve into something more robust, such as a voice recognition based system.

Currently our system is designed to be explicitly told when trips start and begin. In the future as more resources are added to the system it may become possible to have the system intelligently predict the beginning and end of trips.

The leg data that is collected for the system currently includes a limited number of fields, such as speed, date, time, and beginning & ending locations. As more systems are brought into the car and resources increase, the LogBook can be potentially expanded to capture more data about a trip.

The current storage limitations on the card limit the number of trips that can be accumulated on a card. As SmartCard technology improves this limitation should disappear.

3.3.9 Physical Environment

The target equipment, SmartCard and the SmartCard reader, will operate in the car and on an external system. The SmartCard, external system, and SmartCard readers will be in different locations. The SmartCard will provide the mechanism for transferring data between the car and external system. Readers would typically exist in the car, and at the location of the external system.

3.3.10 Security Issues
The access to the leg data by the user of the card must be controlled. The normal driver should not be able to see the legs of a trip, but only the trips itself. Furthermore, the driver should only be able to see his/her trips and be denied access to other trip information in the vehicle. The non-authorized driver should not be able to modify any of the trip information. This includes additions and/or deletions from the logs. Therefore, there will be two types of users, the administrators and users. The administrator will be able to grant the users certain access rights on viewing of the trips. Physical security is an issue because the user may have downloaded some special program on the card which may try to infiltrate the system. Another issue is whether an infiltrator might physically remove the storage medium from the car, in which case the log data may need to be encrypted when it needs to go on the permanent medium.

3.3.11 Resource Issues

The resources in the LogBook use the card as a data transfer medium and will be backed up on the external system whenever the card is inserted to the card reader attached thereof. The LogBook system will be installed along with the rest of the JAMES system, most likely at the factory with the rest of the car. The administrator will be free to administrate the logs as they see fit. Data on the external system will be considered to be the master data; if an administrator modifies data on the external system and the resulting data conflicts with newly arrived data from the vehicle or car, the external system data will be used.

3.4 Constraints

Java is the programming language of choice for this project, and both the car's computer and the desktop computer should be able to run the language in its entirety. The computer on the card, however, is not nearly as powerful and can only handle a small fraction of the original language's resources.

The card's version of Java carries the following restrictions. It can't handle threads, exceptions, static initializers, data types larger than short ints, multi-dimensional arrays, and arrays of user defined types (classes, etc.).

Other restrictions appear because of the limited size of the Java stack. The stack can hold 32 bytes (possibly only 16 in older cards). With such a limited stack space the function call tree must be kept very shallow, inheritance is not advisable, and the number of variables is restricted. This will hurt the JAMES team's ability to create an elegant design for the java cardlet.

There is an OS library on the card, and while a developer should be able to use other libraries (except threads). It's not recommended that you do much inheritance.

Development environment is only restricted by target platform. Regular Java tools can be used, but Schlumberger's makesolo program must be included in the compilation cycle to create an executable cardlet.

3.5 System Model

The LogBook system is divided into three parts. The part of the LogBook that runs on the car's onboard computer, the part that runs on the card itself (ie. the cardlet), and the external applications for managing LogBook data.

3.5.1 Scenarios

- Joe has to go on a business trip to Philadelphia. He gets into his Mercedes M-Class car, inserts his SmartCard, and selects "Begin Trip" from the dashboard touch screen's LogBook menu. Near Harrisburg he stops at a rest area for food and gas. When he gets back in the car, the JAMES system asks him if he is continuing his current trip, starting a new one, or done recording. He selects "Continue." At his Philadelphia hotel, he selects "End Trip" and parks. He removes his SmartCard and brings it with him to his room, where he inserts it into the PCMCIA card reader in his laptop. The LogBook Assistant updates his private database of business trips. The next day Joe drives home, once again updating his private trip database on arrival. The LogBook Assistant allows him to generate an expense voucher for his company, and to erase the SmartCard so it may be used on his next business trip.

- Otto is the manager of a limousine service in Stuttgart. His employee drivers use SmartCards to authenticate themselves to the JAMES system. At the end of each day, Otto uses his Administrator-level SmartCard to transfer all trip records from his cars to his central database. There he views each trip and leg traveled by each vehicle, correcting any errors made by his staff by adding, removing, and rearranging legs within trips. Since one of his rivers is leaving the company, Otto deletes him from the database. He adds a user
representing the new driver he has hired as a replacement, giving the user access rights appropriate to his status within the company.

- A family wants to keep good records of their expenditures. Whenever they take a trip in the family car, they use their SmartCard and the LogBook Assistant to record the usage and cost of gas. They update the database on their home computer weekly, and every April they have the LogBook Assistant generate a tax form which describes their costs.

### 3.5.2 Use Case Models

#### 3.5.2.1 Actors

- User - Driver or administrator of the car.
- SmartCard - Electronic card used to facilitate access to JAMES services.
- Vehicle Subsystem - Hardware/software system providing access to conventional (non-JAMES) vehicle components.
- Authorization Subsystem - JAMES component which establishes user's identity for data security purposes.
- Database - Component used to store all LogBook data, including trip/leg information and user access rights.

#### 3.5.2.2 Use Cases

##### 3.5.2.2.1 Use Case Mappings

This section describes the mapping between the use cases provided by the client (in `dev/logbook/model/ud/Logb4.mdl` in Perforce) and the use cases presented in this RAD.

The following are the areas where we believe the use cases coincide.

- The client use cases contain DoingATrip use case and a LogbookTransactionManager. Our Start, End, Suspend/Resume maps to both of these use cases. Our use cases handle part of the job of handling the users input and talking with the outside world (including that of retrieving info from the car). However, we go into further detail, in that we describe the associations between legs and trips in our use cases, where as the client does not present this idea. This model of representing trips in terms of legs has been accepted by the client, therefore this is how we have our use cases.

- The client use cases contain a MonitorTrip use case. This should map to our RetrieveTripInfo use case. RetrieveTripInfo also handles the job described in the client use cases as LogbookTransactionManager. Again, we go into more detail, in that we allow for saving to the card, filtering (viewing only certain trips), and deleting of trips (if authorized). However, these are just different views, or ways to monitor data contained in the logbook.

- Overall, we merge the two use cases: DoingATrip and LogbookTransactionManager then separate it into its various components (Start, End, Suspend/Resume). Therefore each of our use cases handle data gathering (from the user) and data storage tasks.

The following are the areas where we believe the use cases do not directly coincide, however, we believe the issues have been discussed with the client, and are therefore presented differently in our use cases.

- The LogbookTransactionManager & InfoBaseLogbook implies that the logbook would be responsible for the actual storage of trips. However we believe that this duty has to be fulfilled by an external storage subsystem or service, therefore we represent this with both the Card and Database actors which are external to our system.

- The client has the carInfoLogbook and FurtherAssistantInfoLogbook actors using our system. However, at no time do we believe that anyone has approached us with specific data that would have to be offered as a service. In a comparison with the Travel Subsystem we found the overlap in the systems to be minimal because that subsystem deals with trips in terms of milestones which are determined in advance, while this subsystem deals with trips in terms of legs which are recorded as they happen (with generally non-predictable data). It does not appear that other subsystems have need of the data we record or services we might make available.

- The client inherits from the Driver class at Corporate User and a Normal Driver. It was our understanding that a card would be specifically geared to either business or personal use, and that this would be represented in the leg object. Therefore, we have User and Authorized Drivers as subclasses of the base Driver class.
3.5.2.2 Onboard Computer Use Cases
Figure 3.1 This is a use case diagram for the onboard system.

Use Case Name: Delete Trips From Vehicle
Actors: SmartCard, Vehicle, Authorization
Entry Condition: SmartCard is inserted into card reader and belongs to vehicle administrator
Flow of events:
- Given a list of trips to delete
- While the list is not empty, repeat the rest of the use case:
  - Delete the trip from the database
  - Remove the trip from the list and the display
Exit Condition: No more trips are selected
Special Requirements:

Use Case Name: End Current Leg
Actors: Driver, Vehicle
Entry Condition: Vehicle ignition is on
Flow of events:
- Driver presses "End Trip" button
- If there is a current trip (active or paused), end the current trip
Exit Condition: Current trip is set to 'none'
Special Requirements: None

Use Case Name: End Current Trip
Actors: Vehicle, Database
Entry Condition: Current trip is not 'none'
Flow of events:
- Write current trip to database
- Set current trip to 'none'
- Begin new leg
Exit Condition: Current trip is 'none' and there is a new leg
Special Requirements: None

Use Case Name: End Trip
Actors: Driver, Vehicle
Entry Condition: Vehicle ignition is on
Flow of events:
- Driver presses "End Trip" button
- If there is a current trip (active or paused), end the current trip
Exit Condition: Current trip is set to 'none'
Special Requirements: None

Use Case Name: Filter Trips
Actors: Vehicle, SmartCard, Authorization
Entry Condition: SmartCard is inserted into card reader, Authorization data is available (driver or administrator)
Flow of events:
- Get authorization data from SmartCard
- If card belongs to car administrator, return all trips stored
- If card belongs to car driver, return a list of all trips in which the driver drove at least one leg
Exit Condition: Filtered list of trips is returned
Special Requirements: None

Use Case Name: Resume Trip
Actors: Vehicle
Entry Condition: Current trip is 'none' and suspended trip is not 'none'
Flow of events:
- Set current trip to suspended trip
- Set suspended trip to 'none'
- Begin new leg

Exit Condition: Current trip is starting suspended trip and suspended trip is 'none'

Special Requirements: None

Use Case Name: **Retrieve Trip Info**

Actors: User, SmartCard, VehicleDatabase

Entry Condition: SmartCard is inserted in vehicle card reader, Authorization data is available (car driver or administrator)

Flow of events:
- User clicks 'Retrieve Logbook Info' button (or an analogous trigger)
- Onboard computer filters list of trips stored to determine which trips are visible and what operations are available on them
- Create new UI window to display trip information and operations
- Computer displays list of trips, with operations permitted, in new UI window
- (*) User selects zero or more trips from list
- User selects operation to perform on list of trips
- If operation is 'exit', do nothing else (close new UI window)
- If operation is 'save to card', save trips to card
- If operation is 'delete', delete trips from vehicle storage
- Return to step (*)

Exit Condition: None

Special Requirements: None

Use Case Name: **Save Trips To Card**

Actors: Vehicle, SmartCard

Entry Condition: SmartCard is inserted into card reader

Flow of events:
- Given a list of trips to store
- While the list is not empty, repeat the rest of the use case:
  - If there is insufficient space on the SmartCard to store the first trip in the list, abort (returning current list)
- Store the first trip in the list to the SmartCard
- Remove the first trip from the list

Exit Condition: The list of trips which did not fit on the SmartCard is returned; if all trips did fit, then return list is empty

Special Requirements: None

Use Case Name: **Start New Leg**

Actors: Vehicle, Database

Entry Condition:

Flow of events:
- Get state data from vehicle
- If current leg is not 'none':
  -- set current leg end state to retrieved state
  -- write current leg to database
- Set current leg to a newly initialized leg
- Set current leg start state to retrieved state

Exit Condition: New leg is initialized and set to be current

Special Requirements: None

Use Case Name: **Start New Trip**

Actors: Vehicle

Entry Condition: Current trip is 'none'

Flow of events:
- Set current trip to a newly initialized, empty trip
- Begin new leg

Exit Condition: Current trip is not 'none' and there is a new leg

Special Requirements: None

Use Case Name: **Start Trip**
Actors: Driver, Vehicle
Entry Condition: Engine is on
Flow of events:
- Driver presses "Start Trip" button
- If there is a current trip (active or paused), end the current trip
- Create a new trip object
Exit Condition: New trip has been created and set as active trip
Special Requirements: None

Use Case Name: **Start Vehicle**
Actors: Driver, Vehicle
Entry Condition: Vehicle ignition is off
Flow of events:
- Driver turns vehicle ignition to on
- Logbook is notified
- Set current leg to 'none'
- A new leg is started
Exit Condition: Current leg is set to new leg
Special Requirements: None

Use Case Name: **Stop Vehicle**
Actors: Driver, Vehicle
Entry Condition: Vehicle ignition is on
Flow of events:
- Driver turns off vehicle ignition
- Logbook is notified
- If previous leg was ended within cutoff criteria of current state, set that leg's end information to current state and erase current leg; otherwise, end current leg
Exit Condition: Current leg is set to 'none'
Special Requirements: None

Use Case Name: **Suspend/Resume Trip**
Actors: Driver, Vehicle
Entry Condition: Vehicle ignition is on
Flow of events:
- Driver presses "Suspend Trip" button (which may currently be labeled "Resume Trip")
- If there is a current trip:
  -- If it is active, suspend it
  -- If it is suspended, resume it
- If no trip is in progress, do nothing
Exit Condition: If there is a current trip, it is suspended if it was previously active and vice versa
Special Requirements: None

Use Case Name: **Suspend Trip**
Actors: Vehicle
Entry Condition: Current trip is not 'none' and suspended trip is 'none'
Flow of events:
- Set suspended trip to current trip
- Set current trip to 'none'
- Begin new leg
Exit Condition: Current trip is 'none' and suspended trip is starting current trip
Special Requirements: None

3.5.2.2.4 External Application Use Cases
Figure 3.2 This shows the use case diagram showing how the use cases for the external system relate to each other.

Use Case Name: **DisplayData**
Actors: User, Database
Entry Condition: Data is available.
Exit Condition: Termination of dialog.
Special Requirements: User has appropriate access rights.
Flow of events:
- DisplayData menu item invoked.
- User prompted for filter(s).
- Data filtered and displayed.
- User may invoke ModifyData or ExportData use cases.
- Dialog terminated.

Use Case Name: **ExportData**
Actors: User
Entry Condition: Data is being displayed.
Exit Condition: Termination of dialog.
Special Requirements: none
Flow of events:
- ExportData menu item invoked.
- User prompted for export format and destination.
- Data exported.
- Dialog terminated.

Use Case Name: **ModifyData**
Actors: User
Entry Condition: User is an administrator, data is being displayed.
Exit Condition: Termination of dialog.
Special Requirements: User has appropriate access rights.
Flow of events:
- ModifyData menu item invoked.
- User prompted for changes.
- User prompted for confirmation of changes.
- Database updated.
- Dialog terminated.

Use Case Name: **SetAccess**
Actors: User, Database
Entry Condition: User is an administrator.
Exit Condition: Termination of dialog.
Special Requirements: none
Flow of Events:
- SetAccess menu item invoked.
- List of known users and their current access rights displayed.
- Administrator may select a user and modify their access rights, create a new user, or delete an existing user.
- Database updated.
- Dialog terminated.

Use Case Name: **UpdateDatabase**
Actor: Database, User, SmartCard
Entry Condition: Database on the external system exists.
Exit Condition: Database on the external system is updated with the data from the SmartCard.
Special Requirements: Card reader attached to PC, card in card reader
Flow of Events:
- User chooses the load data from card option
- Trip data from the card is loaded into memory
- Trip data is inserted into the database
- Database updates itself with the new data.
- Success/Error message is returned.

Use Case Name: **WipeCard**
Actors: User, SmartCard
Entry Condition: Card is in the reader.
Exit Condition: Task completed.
Special Requirements: Card reader attached to PC, card in card reader, user has appropriate access rights.
Flow of events:
- WipeCard menu item invoked.
- Data on card erased (only trip data, not authorization information).
- Task completed.

### 3.5.3 Object Models

#### 3.5.3.1 Data Dictionary

- **Authorization** - a subsystem which uses information on the SmartCard to identify the rights of the user (SmartCard holder) as Driver or Administrator.
- **LogBook** - Cardlet/application system which allows the driver of a car or the administrator of a fleet of cars to maintain travel records which include the duration, distance, and cost of auto trips.
• Current Trip - The trip being recorded.
• Paused Trip - The trip that was being recorded until the user explicitly paused recording.
• Leg - The basic unit of LogBook travel measurement, bounded in time by the starting of the car, stopping of the car, or changing of the driver.
• Trip - A collection of legs, as defined by the user. Generally, these will be consecutive legs on the path to a single destination.
• Database - The repository for LogBook information, which is stored in units of legs. The database may be queried in units of legs or trips, filtered by date, driver, or vehicle used.
• Document - Collection of multiple trips.
• Administrator - Manager of a fleet of cars. Authorized to obtain information about any driver or vehicle in the fleet. Authorized to set and change access rights for other users. In the case of a privately owned car, the driver and the administrator will generally be the same person.
• User - One who drives a vehicle. Authorized to obtain information about his or her own trips in a vehicle. May be authorized to obtain more information, at the discretion of the fleet administrator.
• SmartCard - Credit-card-like appliance which simplifies the processes of authentication and authorization, recording, and data transfer in the JAMES system.
• Annotation - Textual comments associated with a leg, made by the user.
• Filter - A set of contraints on the data set. If the data set were trips, one filter would be to get the list of trips between two dates.

3.5.3.2 Class Diagrams

3.5.3.2.1 Document Class Diagram

![Document Class Diagram](image-url)
**Figure 3.3** This diagram shows the Document class which is a collection of trips. It also contains those legs which are not part of any trip.

### 3.5.3.2.2 Leg Class Diagram

**Figure 3.4** This diagram shows the Leg class which stores data pertaining to the beginning and ending of a leg including the driver of the leg, gas usage, beginning and end location, maximum speed, mileage, and times.

### 3.5.3.2.3 Main Class Diagram
Figure 3.5  This diagram shows the main classes and the relationships between them including the document and its relationship with the application in the vehicle and on the external system and with the SmartCard itself.

3.5.4 Dynamic Models

3.5.4.1 Sequence Diagrams

Start Vehicle
Figure 3.6 The user is starting the vehicle.

Start Trip
Figure 3.7 The user decided to start the trip.

End Trip

Figure 3.8 The user decided to end the trip.
End Current Trip

Figure 3.9 The current trip is ended.
Figure 3.10 This shows how a filter on the list of trips that is displayed is applied.

This sequence diagrams shows that the user creates a filter. A filter is an object and the display can have multiple filters that combine the filters to produce the data. After the filter is added, the display is refreshed.
Figure 3.11 Sequence diagram whereby the user chooses to export the current data set.

The user chooses to export in this sequence diagram. The export object allows flexible export methods that could be subclassed into such objects as print, save to database format, etc.
Figure 3.12 Modify data that allows modifications on the properties of the trips and also delete.
Figure 3.13 An authenticated user changing the access rights of a user.
Figure 3.14 The user is updating the database by retrieving the information from the SmartCard.
3.5.4.2 State Diagrams

Figure 3.15 The user wiping the logbook information from the SmartCard.

Figure 3.16 The state diagram that shows how the trip mechanism works. Initially, there is no trip active for the logbook assistant. The user will start a trip, and legs will be added while the
trip is active. The user may suspend a trip. If there is a currently suspended trip, then the new trip and end trip functions apply to that suspended trip, or resume the trip and add new legs to it.

3.5.5 User Interface - Navigational Paths and Screen Mockups

Figure 3.17 External Application Screen Mockup.