ROAD TRIP PLANNER

A
Case Study Report
Submitted in partial fulfillment of the
Requirements for the Course of

SOFTWARE ENGINEERING LAB

IN

BE 2/4 (IT) IV-SEMESTER

By

Akshitha Doodala
1602-20-737-003

Shruthi Kovvur
1602-20-737-036

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, **Akshitha D** and **Shruthi K,** bearing hall ticket number, **1602-20-737-003** and **1602-20-737-036,** hereby declare that the Case study report entitled **"Active Park Assist"** under the guidance of **Ms. Sree Laxmi,** Assistant Professor, Department of Information Technology, VCE, Hyderabad is submitted in partial fulfillment of the requirement for the course of **Software Engineering - Lab** in BE 2/4 (IT) IV- Semester.

This is a record of bonafide work carried out by me and the Design embodied in this project report has not been submitted by any other.

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-500 031

Department of Information technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **"Road Trip Planner"** being submitted by **Akshitha Doodala** and **Shruthi Kovvur,** bearing **1602-20-737-003** and **1602-20-737-036,** in partial fulfillment of the requirement for the course of **Software Engineering - Lab** in BE 2/4 (IT) IV- Semester is a record of bonafide work carried out by him under my guidance.

Signature
Internal Examiner

Signature
External Examiner

ABSTRACT

Road Tour Planner (RTP) is a web application designed to help you plan road trips. You can enter a starting point and a destination, and it will suggest potential routes with pit stops along the way. In order to make a road trip successful, this web service aims to minimize the number of applications or services the user would have to switch between. Our users can pre-plan their trips before they hit the road. Additionally, the application will prepare an itinerary for the user on a daily basis for the duration of their vacation, not just routing options. The application will use a variety of APIs such as Navigation APIs, Location APIs, Reviews APIs, and Current Events APIs. It will also leverage AI technology to create more personalized itinerary plans based on user input.

Software Requirements Specification

(Road Trip Planner)

1. Problem Statement: To collect software requirements and create software requirement specification.

SOFTWARE REQUIREMENTS: Ms Word.

1. Introduction

1.1 Purpose Basic Description of Problem

This document describes the software functionalities and requirements for a road trip planner web service, which will give the user detailed information about the trip they are planning. We will also discuss system constraints, interfaces, and interactions with external applications.

1.2 Scope

Road Tour Planner is a web-based application that lets you plan exciting road trips. Once the user enters a starting and destination point, the web application suggests several road trip alternatives along with restaurants, hotels, and activity options tailored to the user's preferences. The users will be able to view road suggestions for their trips, create a new tour based on starting and destination points, and log in to save their trip information

1.3 Definitions, Acronyms, and Abbreviations

API - Application Programming Interface

ETA - Estimated Time of Arrival

HTTPS - Hyper Text Transfer Protocol Secure

RTP - Road Tour Planner

SSL - Secure Socket Layer

UML - Unified Modelling language

1.4 References

1. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specification

2. J. Peters, and W. Pedrycz, Software Engineering – An Engineering Approach. New York, NY: Wiley, 2000.

3. D. P. Gilliam, T. L. Wolfe, J. S. Sherif, and M. Bishop, “Software Security Checklist for the Software Life Cycle,” in Proc. WETICE’03, 2003, pp. 243-248.

4. A. D. Rubin, “Security Considerations for Remote Electronic Voting,” CACM, vol. 45, pp. 39-44, Dec. 2002.

2. Overall Description

2.1 Objective:

Road Tour Planner (RTP) is a web application designed to help you plan road trips. You can enter a starting point and a destination, and it will suggest potential routes with pit stops along the way. In order to make a road trip successful, this web service aims to minimize the number of applications or services the user would have to switch between. Our users can pre-plan their trips before they hit the road. Additionally, the application will prepare an itinerary for the user on a daily basis for the duration of their vacation, not just routing options. The application will use a variety of APIs such as Navigation APIs, Location APIs, Reviews APIs, and Current Events APIs. It will also leverage AI technology to create more personalized itinerary plans based on user input.

Road Tour Planner (RTP) is a web application designed to help you plan road trips. You can enter a starting point and a destination, and it will suggest potential routes with pit stops along the way. In order to make a road trip successful, this web service aims to minimize the number of applications or services the user would have to switch between. Our users can pre-plan their trips before they hit the road. Additionally, the application will prepare an itinerary for the user on a daily basis for the duration of their vacation, not just routing options. The application will use a variety of APIs such as Navigation APIs, Location APIs, Reviews APIs, and Current Events APIs. It will also leverage AI technology to create more personalized itinerary plans based on user input.

3. External Interface Requirements

3.1 User Requirement

Need for an application that makes communicating easy and comfortable. An application that helps user create plan for their road trip .

Need for an application that is easy to use and widely available and hence a web application Handling all functions done with organization in a computerized manner.

3.2 Functional Requirement

Login: login with your name and password

Create Trip: Creates Trip

Edit Trip: Helps you edit your trip plan

Share Trip: You can save your trip.

3.3 Non-functional Requirement

Usability: This website has appropriate user interface and adequate information to guide the user to use the website.

Portability: The website is portable as it is online website running across the net

Flexibility: It is very flexible

• Security: This website provides user and authentication so that only the legitimate user is allowed to use

the website

Maintainability: This website is capable to secure the data and easily retrieve the data.

Scalability: This system can further modified in future.

UML Introduction:

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However, it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modeling systems, ranging from enterprise information system to distributed web-based application and even to real time embedded system**.**

UML includes nine diagrams:

Use Case Diagram: To illustrate the user intersection with the system

Class Diagram: To illustrate logical structure.

Object Diagram: To illustrate the physical structure of the software.

Deployment Diagram: To shows the mapping of the software- hardware.

Interaction Diagram: To show the collaboration of group of objects.

Sequence & Collaboration: To illustrate behavior.

State Chart & Activity Diagram: To illustrate flow of events

Use-case Diagram: A use case diagram shows interactions of actors with a system in terms of functions called use-case. A use case is description of a functionality that the system provides. The actors are external to the system but who interacts with thesystem; they may be external persons, external system and hardware.

Class Diagram: A class diagram shows the static structure of classes in the system.

The classes represent the "things" that are handled in the system. Class can be related toeach other in a number of ways: associated, dependent, specialized or packaged. Class represents attributes and operations.

Object Diagram: An object diagram is a variant of the class diagram and uses almost identical notation. The difference between the two is that an object diagram shows a number of object instances of classes, instead of actual classes. Objects are written in rectangular box with their names. object diagrams are written in rectangular box with their names. Object diagrams are not important as class diagrams but show the actual instances of a class and the relationships.

State Diagram: A state diagram is typically a component to the description of a class. It shows all possible states that the object of the class can have, and which events causethe state to change. An event can be another object that sends a message to it. A change is called a transition. State diagrams are not drawn for all classes, only for those that have a number of well-defined states.

State diagram can also be drawn for the system as a whole.

Sequence Diagram: A sequence diagram shows a dynamic collaboration between a number of objects and shows an interaction between objects. This diagram shows a number of objects with vertical lines that represent object lifeline. Messages passing between objects are shown with arrows. This represents the scenario of functions how they apply.

Collaboration Diagram: It is similar to a sequence diagram however the relationships are only shown. If time or sequence of events is important then sequence diagrams are more relevant.

Activity Diagram: An activity diagram shows a sequential flow of activities. This is typically used to describe the activities performed in an operation.

Deployment Diagram: It represents the processors and devices to develop the system ,it shows deployment view of the system.

Component Diagram: A component diagram shows the physical structure of the code in terms of code components. A component can be a source code component, a binary component or an executable component. A component contains information about logical class or classes it implement .

USECASE DIAGRAM

DEFINITION:

Use case Diagram is a collection of use cases, actors and relationships that exists between them. Use-case diagrams graphically depict system behavior (use cases).

These diagrams present a high-level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

CONTENTS:

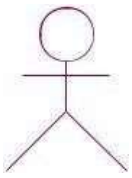
A use-case diagram can contain:

1. Actors ("things" outside the system):

Actors represent system users. They help delimit the system and give a clearer pictureof what the system should do. It is important to note that an actor interacts with but has no control over the use cases.

An actor is someone or something that:

- Interacts with or uses the system
- Provides input to and receives information from the system
- Is external to the system and has no control over the use cases



Actor

2·**Use cases** (system boundaries identifying what the system should do):

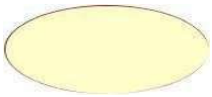
A more detailed description might characterize a use case as:

- a pattern of behavior the system exhibits
- a sequence of related transactions performed by an actor and the system
- delivering something of value to the actor.

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

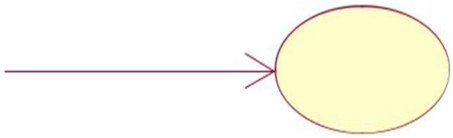
Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.



use case

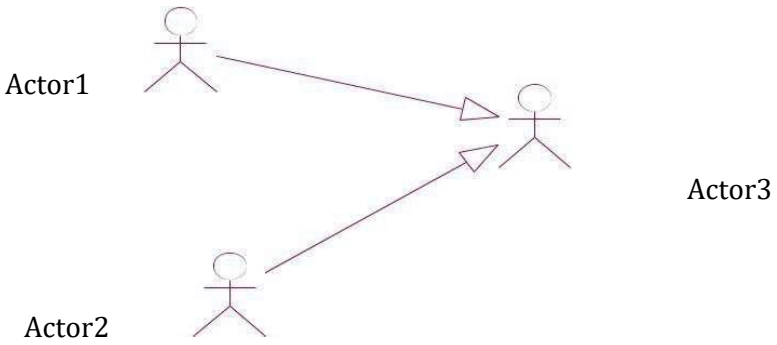
2. Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Association Relationship: An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is an association.



Generalization Relationship:

A generalize relationship is a relationship between a more general class or use case and a more specific class or use case. A generalization is shown as a solid-line path from the more specific element to a more general element. The tip or a generalization is a large hollow triangle pointing to the more general element.



Dependency Relationship:

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning. Typically, on class diagrams, a dependency relationship indicates that the operations of the client invoke operations of the supplier.

DIAGRAM:

AIM: Use Case Diagram for Road Trip Planner.

REQUIREMENTS:

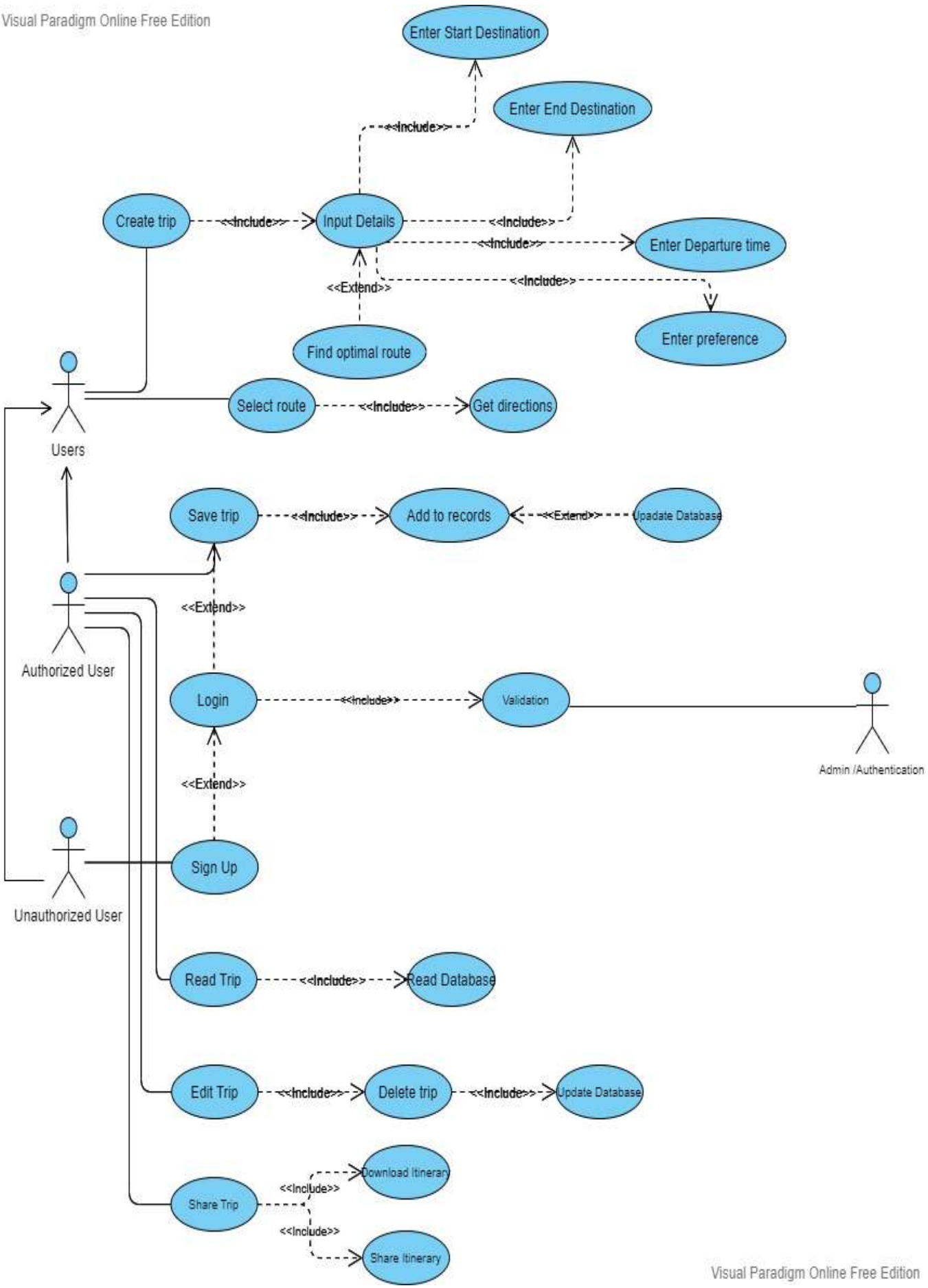
HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

SOFTWARE **:** IBM software

Use cases:

Some of our use cases in our system are

- Login
- Input Details
- Select Route
- Get Directions
- Save Trip
- Delete Trip
- Edit Trip


Visual Paradigm Online Free Edition
Enter Start Destination
Enter End Destination
<<Include>>
Create trip
<<Include>>
Input Details
<<Include>>
<<Include>>
Enter Departure time
<<Include>>
<<Extend>>
Enter preference
Find optimal route
Users
Select route
<<Include>>
Get directions
Save trip
<<Include>>
Add to records
<<Extend>>
Upadate Database
<<Extend>>
Authorized User
Login
<<Include>>
Validation
Admin /Authentication
<<Extend>>
Sign Up
Unauthorized User
Read Trip
<<Include>>
Read Database
Edit Trip
<<Include>>
Delete trip
<<Include>>
Update Database
Share Trip
<<Include>>
Download Itinerary
<<Include>>
Share Itinerary
Visual Paradigm Online Free Edition

CLASS DIAGRAM

DEFINITION:

A class diagram shows the existence of classes and their relationships in the logical design of a system.

A class diagram may represent all or part of the class structure of a system.

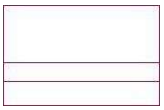
CONTENTS:

A class diagram contains

1. Classes
2. Relationships
3. Interfaces

1) Classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). Aclass is an abstraction of real-world items. When these items exist in the real world, they are instances of the class and are referred to as objects.

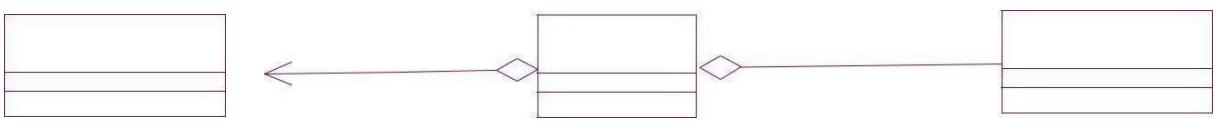
class 

2) Relationships:

Aggregate Relationship

Use the aggregate relationship to show a whole and part relationship between two classes.

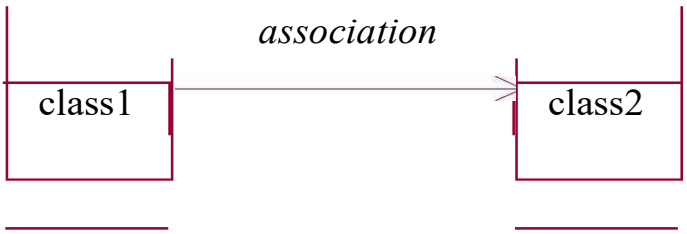
The class at the client end of the aggregate relationship is sometimes called the aggregate class. An instance of the aggregate class is an aggregate object. The class at the supplier end of the aggregate relationship is the part whose instances are contained or owned by theaggregate object.



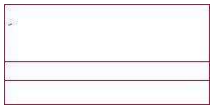
supplier A client Supplier B

Association Relationship

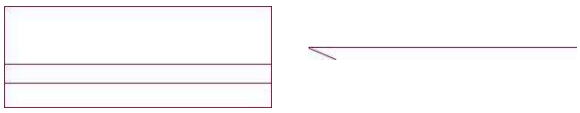
An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. Associations are the most general of all relationships and consequentially the most semantically weak. If two objects are usually considered independently, the relationship is an association



Dependency Relationship

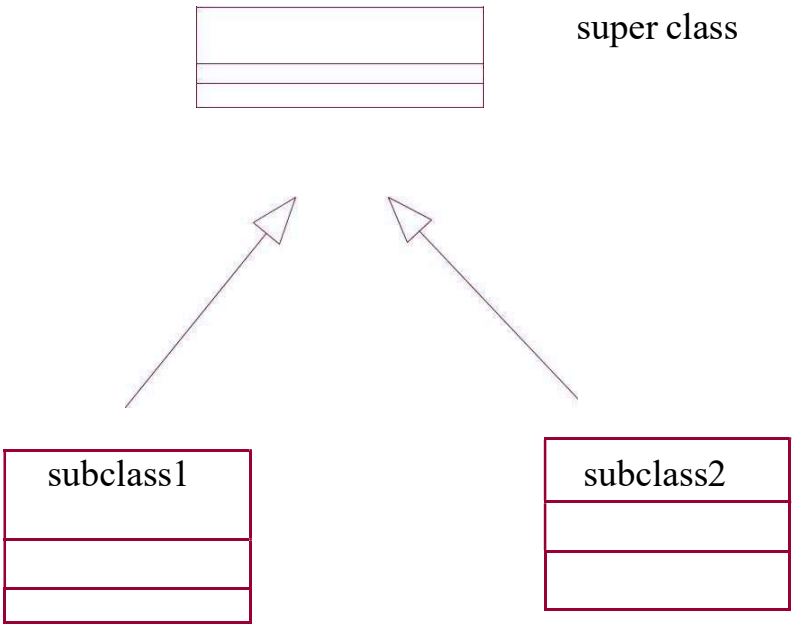


A dependency is a relationship between two model elements in which a change to onemodel element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning



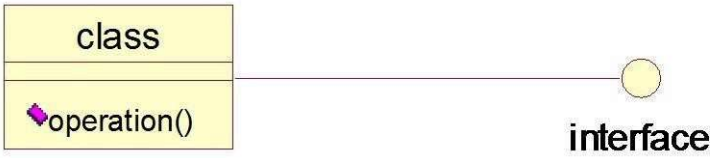
Generalize Relationship

A generalize relationship between classes shows that the subclass shares the structure or behavior defined in one or more super classes. Use a generalize relationship to show a "is-a" relationship between classes.



Realize Relationship

A realization is a relationship between classes, interfaces, components, and packages that connects a client element with a supplier element. A realization relationship between classes and interfaces and between components and interfaces shows that the class realizes the operations offered by the interface.



3) Interfaces

An interface specifies the externally visible operations of a class and/or component, andhas no implementation of its own. An interface typically specifies only a limited part of the behavior of a class or component.

Interfaces belong to the logical view but can occur in class, use case and componentdiagrams.

An interface in a component diagram is displayed as a small circle with a line to the component that realizes the interface:

Adornments

You can further define an interface using the Class Specification. Some class specification fields correspond to adornment and compartment information that youcan display in the class diagram.

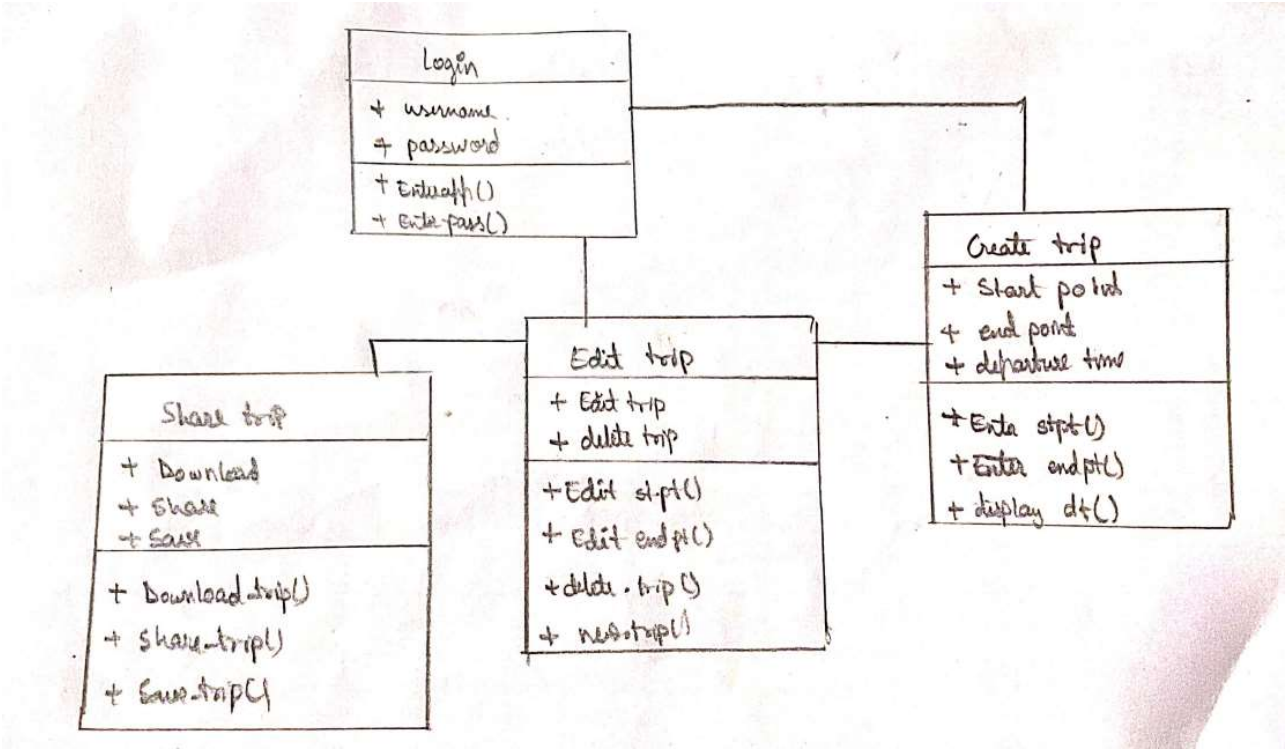
DIAGRAM:

AIM: To draw simple Class diagram for Road Trip Planner.

REQUIREMENTS:

HARDWARE: PIII Processor, 512 MB RAM, 80GB

SOFTWARE: IBM software using Class diagram tools.



SEQUENCE DIAGRAM

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence¾what happens first, what happens next.

Sequence diagrams establish the roles of objects and help provide essential informationto determine class responsibilities and interfaces.

This type of diagram is best used during early analysis phases in design because theyare simple and easy to comprehend.

Sequence diagrams are normally associated with use cases.

Sequence diagrams are closely related to collaboration diagrams and both are alternate representations of an interaction.

There are two main differences between sequence and collaboration diagrams:Sequence

diagrams show time-based object interaction

While collaboration diagrams show how objects associate with each other.

CONTENTS:

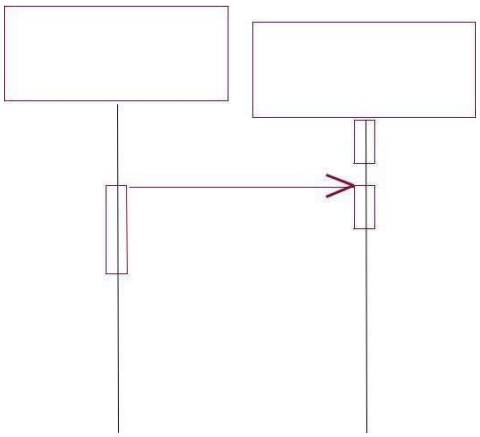
A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

The following tools located on the sequence diagram toolbox enable you to modelsequence diagrams:

Object: An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

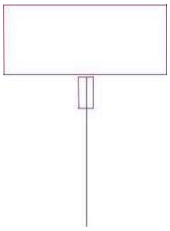
If you use the same name for several object icons appearing in the same collaboration oractivity diagram, they are assumed to represent the same object; otherwise, each object icon represents a distinct object. Object icons appearing in different diagrams denote different objects, even if their names are identical. Objects can be named three different ways: object name, object name and class, or just by the class name itself.

Message Icons: A message icon represents the communication between objects indicating that an action will follow. The message icon is a horizontal, solid arrowconnecting two lifelines together. The following message icons show three different ways a message icon can appear message icon only, message icon with sequence number, and message icon with sequence number and message label.



Each message icon represents a message passed between two objects, and indicates the direction of message is going. A message icon in a collaboration diagram can represent multiple messages. A message icon in a sequence diagram represents exactly one message.

Focus of Control: Focus of Control (FOC) is an advanced notational technique that enhances sequence diagrams. It shows the period of time during which an object is performing an action, either directly or through an underlying procedure. FOC is portrayed through narrow rectangles that adorn lifelines. The length of an FOC indicates theamount of time it takes for a message to be performed. When you move a message vertically, each dependent message will move vertically as well. Also, you can move a FOC vertically off of the source FOC to make it detached and independent. An illustration of a sequence diagram with FOC notation follows:



Message to Self: A Message to Self is a tool that sends a message from one object back tothe same object. It does not involve other objects because the message returns to the same object. The sender of a message is the same as the receiver.

Note: A note captures the assumptions and decisions applied during analysis and design. Notes may contain any information, including plain text, fragments of code, or references to other documents. Notes are also used as a means of linking diagrams. A note holds an unlimited amount of text and can be sized accordingly.

Notes behave like labels. They are available on all diagram toolboxes, but they are not considered part of the model. Notes may be deleted like any other item on a diagram.

Note Anchor: A note anchor connects a note to the element that it affects. To draw anote anchor, place a note on the diagram and connect the note to an element with the note anchor icon.

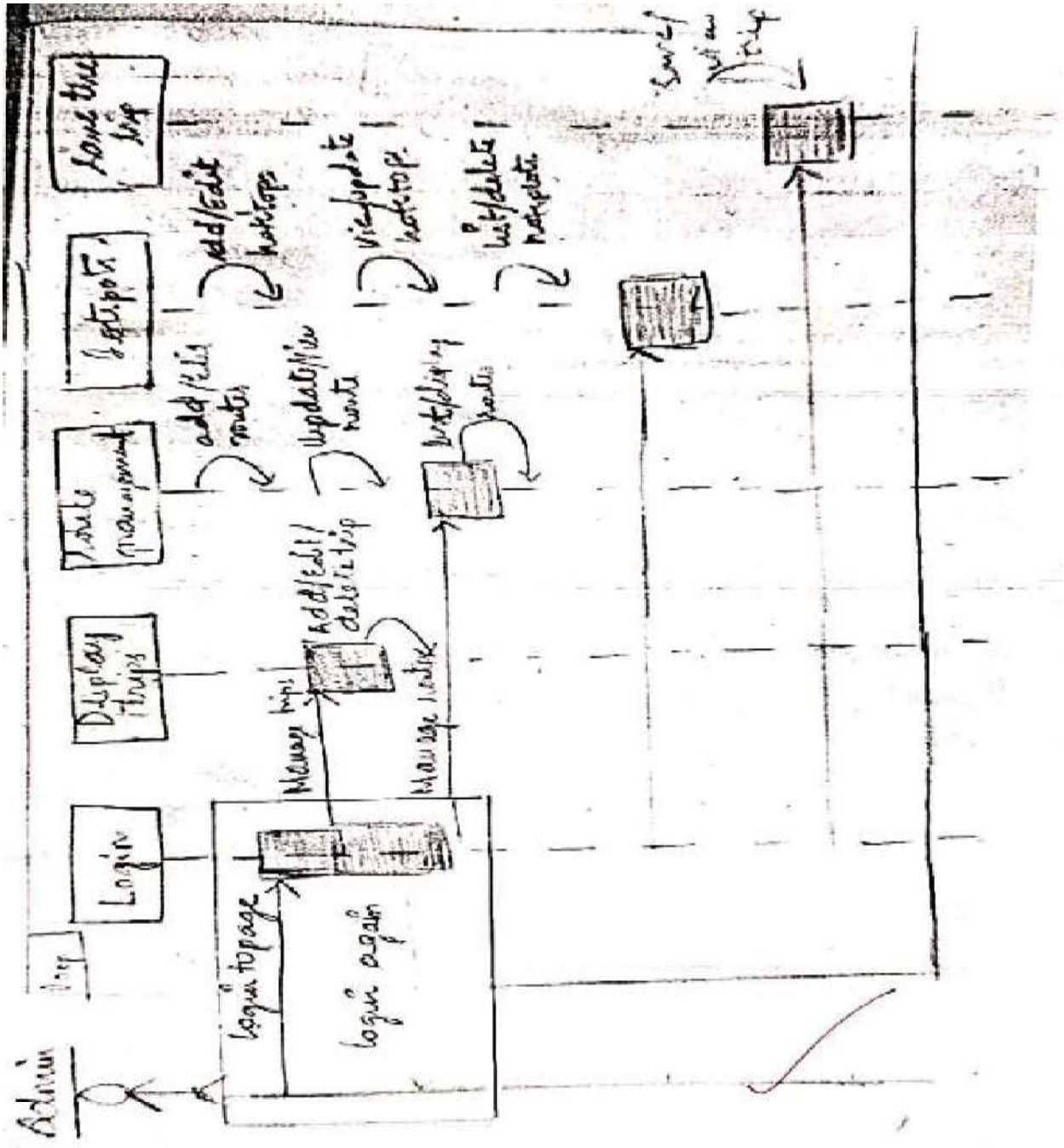
DIAGRAM:

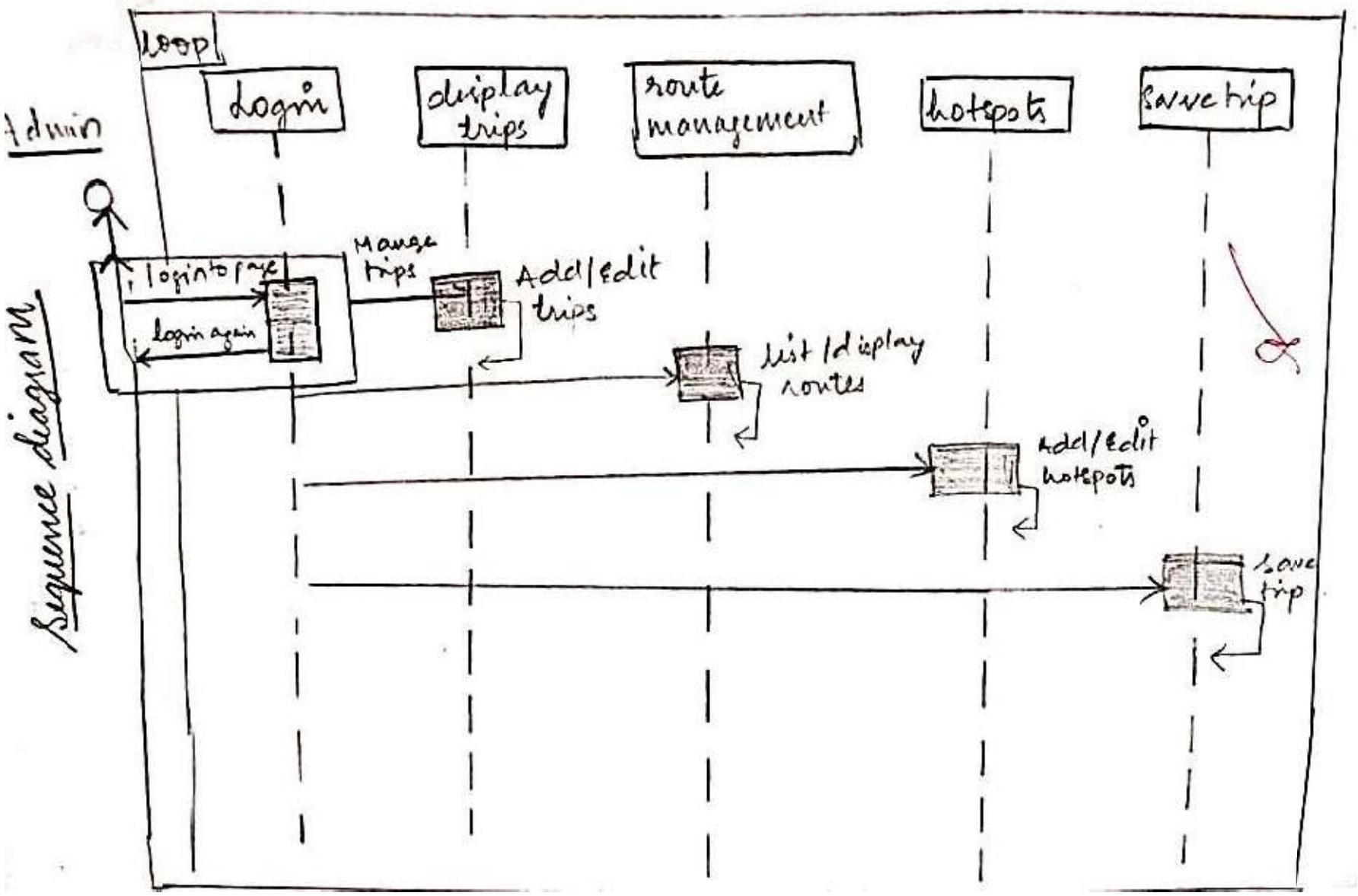
AIM: To draw sequence diagram for Road Trip Planner.

REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB Harddisk.

SOFTWARE **:** In Rational Architecture software using sequence Diagram tool.




Sequence diagram
Admin
loop
login
display trips
route management
hotspots
save trip
login to page
login again
Manage trips
Add/edit trips
list/display routes
Add/edit hotspots
save trip

COLLABORATION DIAGRAM

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time-based sequence.

Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of interactions or structural relationships that occur between objects and object-like entities in the current model.

The Create Collaboration Diagram Command creates a collaboration diagram from information contained in the sequence diagram. The Create Sequence Diagram Command creates a sequence diagram from information contained in the interaction's collaboration diagram. The Go to Sequence Diagram and Go to CollaborationDiagram commands traverse between an interaction's two representations.

Collaboration diagrams contain icons representing objects. You can create one or more collaboration diagrams to depict interactions for each logical package in your model. Such collaboration diagrams are themselves contained by the logical package enclosing the objects they depict.

An Object Specification enables you to display and modify the properties andrelationships of an object. The information in a specification is presented textually. Some of this information can also be displayed inside the icons representing objects in collaboration diagrams.
You can change properties or relationships by editing the specification or modifying
the icon on the diagram. The associated diagrams or specifications are automatically updated.

During: Analysis

Use Collaboration Diagrams: To Indicate the semantics of the primary and secondary interactions.

Design Show the semantics of mechanisms in the logical design of the system.
Use collaboration diagrams as the primary vehicle to describe interactions that expressyour decisions about the behavior of the system.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects aredefined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

If you use the same name for several object icons appearing in the same collaboration oractivity diagram, they are assumed to represent the same object; otherwise, each object icon represents a distinct object. Object icons appearing in different diagrams denote different objects, even if their names are identical. Objects can be named three different ways: object name, object name and class, or just by the class name itself.

If you specify the name of the object's class in the Object Specification, the name must identify a class defined in the model.

Graphical Depiction

The object icon is similar to a class icon except that the name is underlined:
If you have multiple objects that are instances of the same class, you can modify the object icon by clicking Multiple Instances in the Object Specification. When you selectthis field, the icon is changed from one object to three staggered objects:

Concurrency

An object's concurrency is defined by the concurrency of its class.

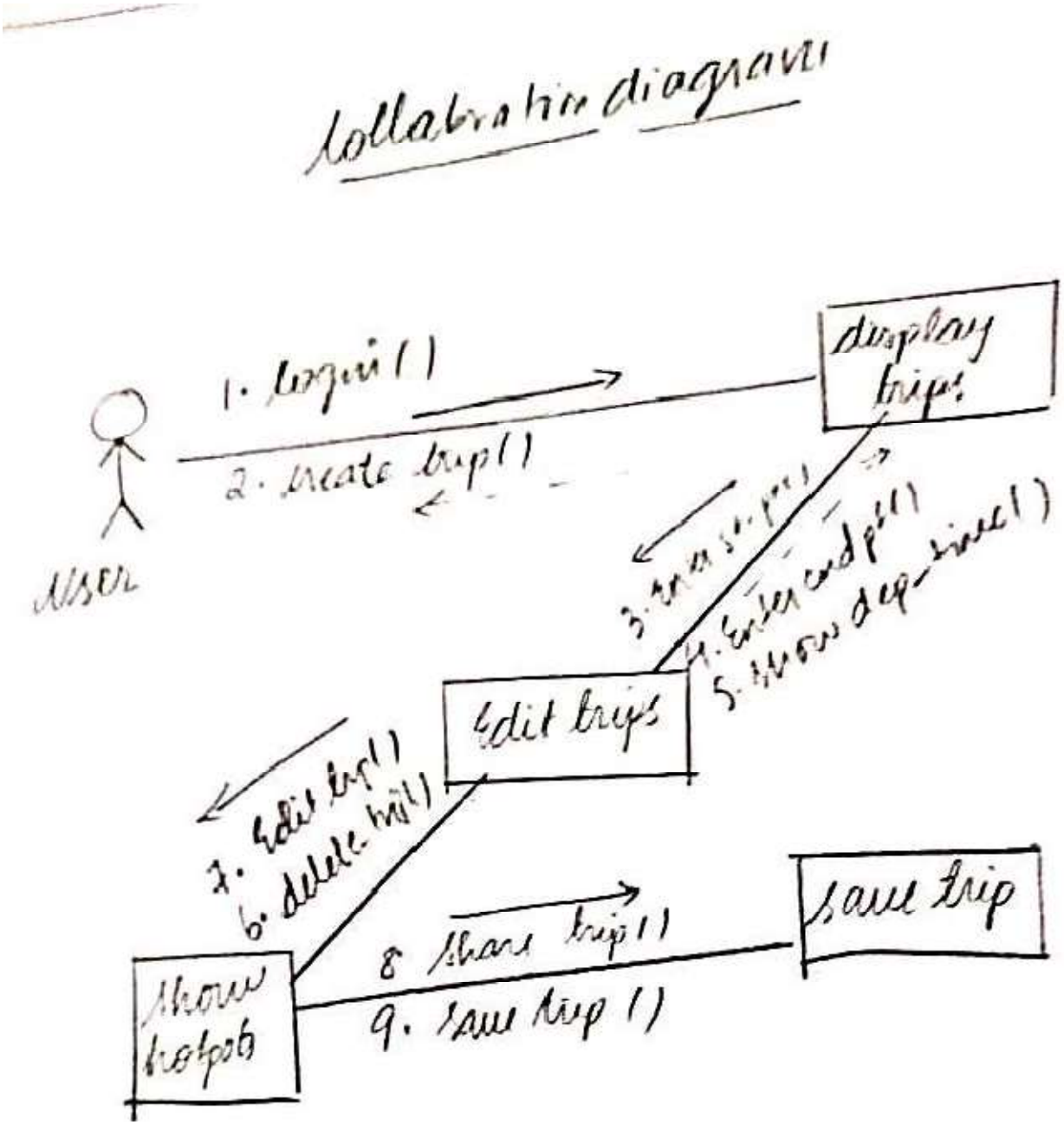
You can display concurrency by clicking Show Concurrency from the object's shortcutmenu. The adornment is displayed at the bottom of the object ic

DIAGRAM:

AIM: To draw Collaboration diagram for Road Trip Planner.
REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

SOFTWARE **:** In Rational Architecture software using Collaboration Diagram tools



ACTIVITY DIAGRAM

DEFINITION:

Activity diagrams provide a way to model the workflow of a business process. You can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity.

An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities. The main difference between activity diagrams and state charts is activity diagrams areactivity centric, while state charts are state centric.

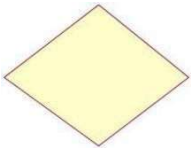
An activity diagram is typically used for modeling the sequence of activities in a process, whereas a state chart is better suited to model the discrete stages of an object's lifetime.

CONTENTS:

Activity Diagram Tools
You can use the following tools on the activity diagram toolbox to model activitydiagrams:

- **Activities:** An activity represents the performance of task or duty in a workflow. It may also represent the execution of a statement in a procedure. An activity is similar toa state but expresses the intent that there is no significant waiting (for events) in an activity. Transitions connect activities with other model elements and object flows connect activities with objects.

Decisions: A decision represents a specific location on an activity diagram or state chart diagram where the workflow may branch based upon guard conditions. There may be more than two outgoing transitions with different guard conditions, but for themost part, a decision will have only two outgoing transitions determined by a Boolean expression



End State: An end state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.

Object: Rational Rose allows objects on activity, collaboration, and sequence diagrams. Specific to activity diagrams, objects are model elements that represent something you can feel and touch. It might be helpful to think of objects as the nouns of the activity diagram and

activities as the verbs of the activity diagram. Further, objects on activity diagrams allow you to diagram the input and output relationships between activities. In the following diagram, the Submit Defect and Fix Defects can be thought of as the verbs and the defect objects as the nouns in the activity diagram vocabulary. Objects are connected to activities through object flows.

Object Flow: An object flow on an activity diagram represents the relationship betweenan activity and the object that creates it (as an output) or uses it (as an input).

Rational Rose draws object flows as dashed arrows rather than solid arrows to distinguish them from ordinary transitions. Object flows look identical to dependencies that appear on other diagram types.

Start State: A start state (also called an "initial state") explicitly shows the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram. You can have only one start state for each state machine because each workflow/execution of a state machine begins in the same place.

Normally, only one outgoing transition can be placed from the start state. However, multiple transitions may be placed on a start state if at least one of them is labeled with a condition. No incoming transitions are allowed.

States: A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event. Each state represents the cumulative history of its behavior.

Swim lanes: Swim lanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swim lanes arevery similar to an object because they provide a way to tell who is performing a certain role. Swim lanes only appear on activity diagrams. You should place activities within swim lanes to determine which unit is responsible for carrying out the specific activity. For more information on swim lanes, look at the swim lane sample.

When a swim lane is dragged onto an activity diagram, it becomes a swim lane view. Swim lanes appear as small icons in the browser while a swim lane views appearbetween the thin, vertical lines with a header that can be renamed and relocated.

Synchronizations: Synchronizations enable you to see a simultaneous workflow in an activity diagram or state chart diagram. Synchronizations visually define forks and joins representing parallel workflow.

Transitions: A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state.

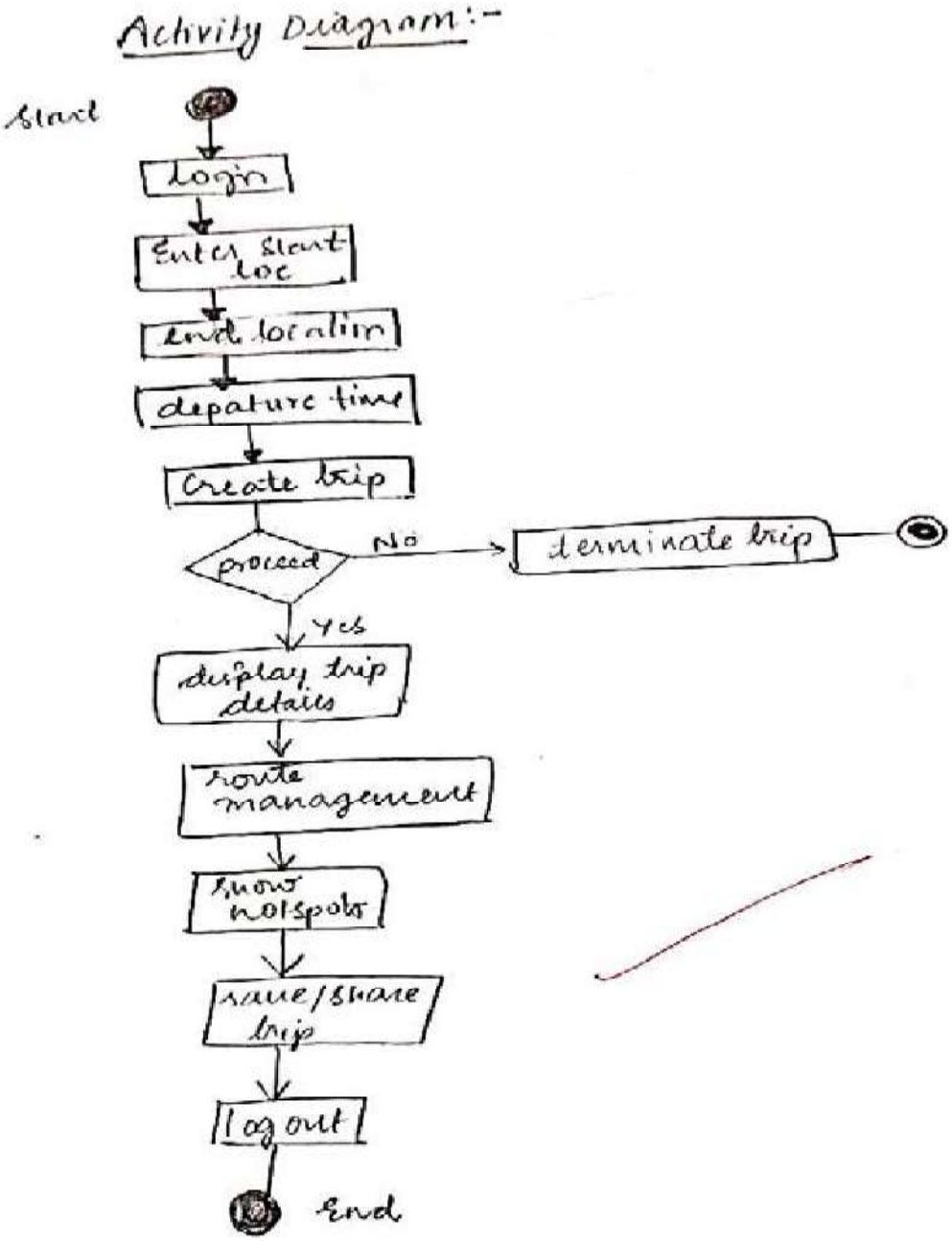
DIAGRAM:

AIM: To draw Activity diagram for Road Trip Planner

REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

SOFTWARE: In Rational Architecture software using Activity diagram tools.



STATE CHART DIAGRAM

AIM: To draw State chart diagram for Road Trip Planner

REQUIREMENTS:

HARDWARE **:** PIII Processor, 512 MB RAM, 80GB

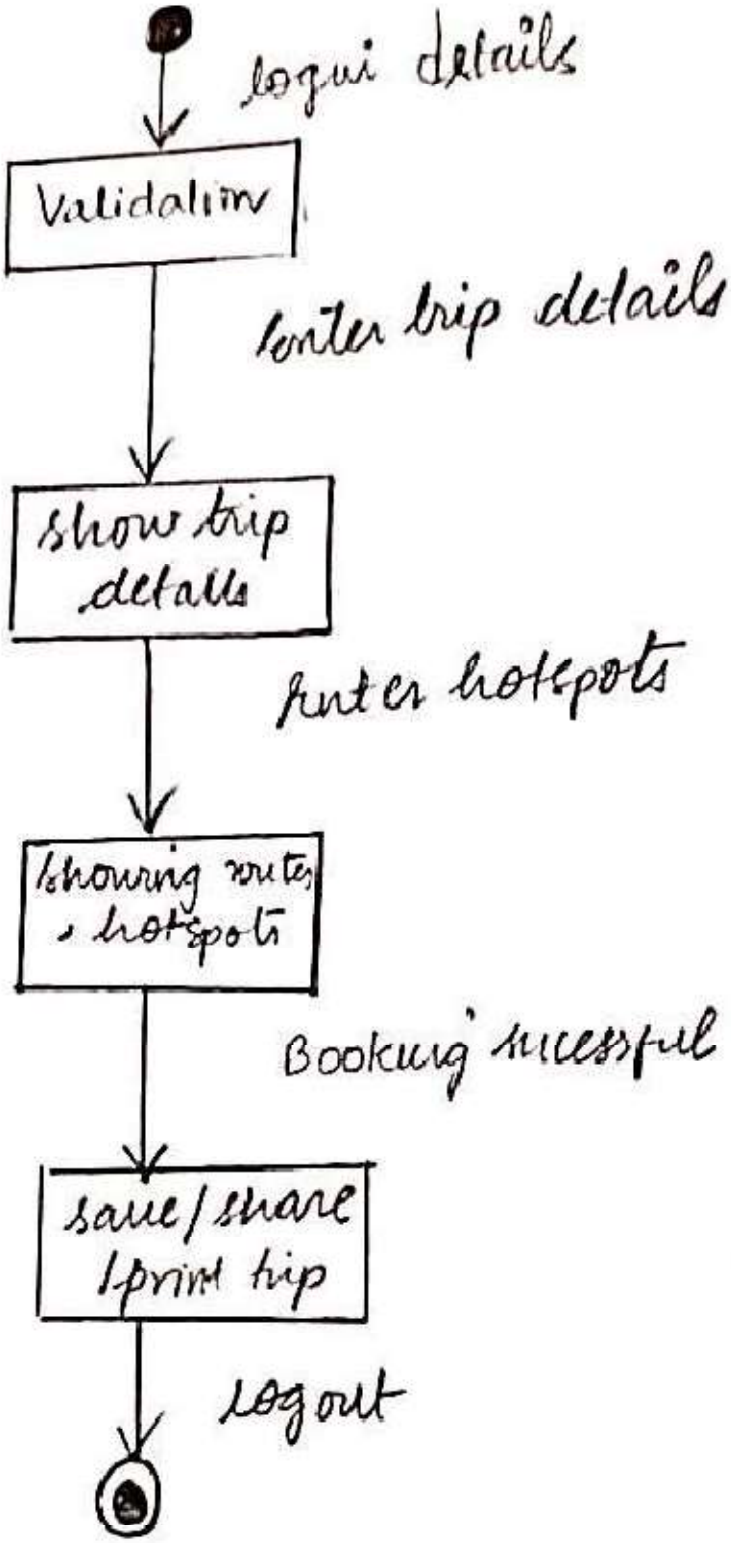
SOFTWARE **:** IBM software Architecture.

DESCRIPTION:

The state chart diagram contains the states in the rectangle boxes and starts in indicated by the dot and finish is indicated by dot encircled. The purpose of state chart diagram is to understand the algorithm in the performing method. State diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered.

- As the name suggests, it describes different states of a Object in a system.
- State Diagrams show the sequences of states an object goes through during its life cycle.
- State diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered.
- The states are specific to a component/object of a system.
- Objects in the system change status in response to events. Used to model dynamic nature of a system.

State diagram

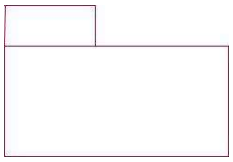


COMPONENT DIAGRAM

Component diagrams provide a physical view of the current model. A component diagram shows the organizations and dependencies among software components, including source code components, binary code components, and executable components. These diagrams also show the externally visible behavior of the components by displaying the interfaces of the components. Calling dependencies among components are shown as dependency relationships between components and interfaces on other components. Note that the interfaces actually belong to the logical view, but they can occur both in class diagrams and in component diagrams.

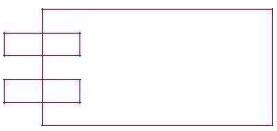
Component diagrams contain:

· **Component packages:** Component packages represent clusters of logically related components, or major pieces of your system. Component packages parallel the role played by logical packages for class diagrams. They allow you to partition the physical model of the system.



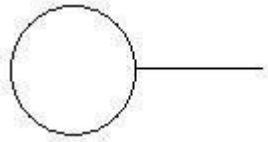
· **Components**: A component represents a software module (source code, binary code, executable, DLL, etc.) with a well-defined interface. The interface of a component is represented by one or several interface elements that the component provides. Components are used to show compiler and run-time dependencies, as well as interface and calling dependencies among software modules. They also show which components implement a specific class.

A system may be composed of several software modules of different kinds. Each software module is represented by a component in the model. To distinguish different kinds of components from each other, stereotypes are used.



Interfaces: An interface specifies the externally-visible operations of a class and/or component,

and has no implementation of its own. An interface typically specifies only alimited part of the



behavior of a class or component.

· **Dependency relationships:** A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning. Typically, on class diagrams, a dependency relationship indicates that theoperations of the client invoke operations of the supplier.

You can create one or more component diagrams to depict the component packages and components at the top level of the component view, or to depict the contents of each component package. Such component diagrams belong to the component package that they depict.

A Component Package Specification enables you to display and modify the properties of a component package. Similarly, a Component Specification and a Class Specification enables you to display and modify the properties of a component and an interface, respectively. The information in these specifications is presented textually. Some of this information can also be displayed inside the icons representing component packages and components in component diagrams, and interfaces in class diagrams.

You can change properties of, or relationships among, component packages, components, and interfaces by editing the specification or modifying the icon on the diagram. The affected diagrams or specifications are automatically updated.

DEPLOYMENT DIAGRAM

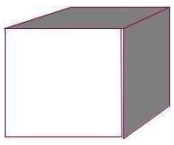
A deployment diagram shows processors, devices, and connections. Each model contains a single deployment diagram which shows the connections between its processors and devices, and the allocation of its processes to processors.

Processor Specifications, Device Specifications, and Connection Specifications enable you to display and modify the respective properties. The information in a specificationis presented textually; some of this information can also be displayed inside the icons.

You can change properties or relationships by editing the specification or modifyingthe icon on the diagram. The deployment diagram specifications are automatically updated.

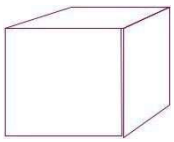
CONTENTS:

Processor:-A processor is a hardware component capable of executing programs.



Devices:

A device is a hardware component with no computing power. Each device musthave a name. Device names can be generic, such as "modem" or "terminal."



Connections:

A connection represents some type of hardware coupling between two entities. An entity is either a processor or a device. The hardware coupling can be direct, such asan RS232 cable, or indirect, such as satellite-to-ground communication. Connectionsare usually bi-directional.

Application Development

Aim: To generate a java code from a UML class diagram for Road Trip Planner.

Software Tools used:

Rational Software Architect

Description:

Software development is a process by which standalone or individual software is created using a specific programming language. Software development may also be called application development and software design. Application development is basically the process of creating a computer program or set of programs that can assist the daily functionalities of the user or business. This is the "original" type of programming. These are 'standard' applications that perform their duties on traditional desktop operating systems, such as Windows, Mac, or Linux. It's often considered a programme, executed on demand by the user, that opens its interface in the confines of the OS that it's running in.

Java, VB.NET, C/C++,C#, Python.

Procedure:

1. Open RSA:File →New→Project
2. Select a wizard as UML Project
3. Enter Project Name
4. Under Create new UML Model →Template →Blank Model
5. Click Finish.
6. Design a class diagram/Select Existing Model and the Save the diagram.
7. Select Blank model
8. Go to menu bar :Modeling →Transfer → New Configuration.

9. Under forward Transformation →IBM Rational Transformations

10. Select UML to Javav5.0→next

12.Under source and target

13.Select your project from source

14.Project name→models→Blank model

15.Click create new target container.

16. Enter Java code Project Name(Passport Java)-->Next

17.Java Settings-->Finish, then(observe Target as Project Name Java)

18.Click Next

19.UML to Java Options-->Next

20.Collections click Next

21.Mapping-->Next

22.Common→Next

23.Select Transformation Options as create source to target relationship.

24.Click Finish.

25. From the Project Explorer ,select Passport UML to JavaCode.tc→Right click→Transform→ UML to Javav5.0

26. From Explorer:- Expand Passport Java Folder→Expand default package→See your classes,Java files

Double Click filename.Java file you can see Java Code.

```
/**
 *
 */

import Login.Sharetrip;

/**
 * @author Administrator
 * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
 */
public class Login {
      /**
       * @author Administrator
       * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
       */
      public class Login_details {
            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public String Username;

            /**
             * @return the Username
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public String getUsername() {
                  // begin-user-code
                  return Username;
                  // end-user-code
            }

            /**
             * @param theUsername the Username to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setUsername(String theUsername) {
                  // begin-user-code
                  Username = theUsername;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private String Password;

            /**
             * @return the Password
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public String getPassword() {
                  // begin-user-code
```

```
                return Password;
                // end-user-code
        }

        /**
         * @param thePassword the Password to set
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public void setPassword(String thePassword) {
                // begin-user-code
                Password = thePassword;
                // end-user-code
        }

        /**
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        private String Forgotpassword;

        /**
         * @return the Forgotpassword
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public String getForgotpassword() {
                // begin-user-code
                return Forgotpassword;
                // end-user-code
        }

        /**
         * @param theForgotpassword the Forgotpassword to set
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public void setForgotpassword(String theForgotpassword) {
                // begin-user-code
                Forgotpassword = theForgotpassword;
                // end-user-code
        }

        /**
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        private CreateTrip class1;

        /**
         * @return the class1
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public CreateTrip getClass1() {
                // begin-user-code
                return class1;
                // end-user-code
        }
```

```
            /**
             * @param theClass1 the class1 to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setClass1(CreateTrip theClass1) {
                  // begin-user-code
                  class1 = theClass1;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private Edittrip class12;

            /**
             * @return the class12
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Edittrip getClass12() {
                  // begin-user-code
                  return class12;
                  // end-user-code
            }

            /**
             * @param theClass12 the class12 to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setClass12(Edittrip theClass12) {
                  // begin-user-code
                  class12 = theClass12;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private CreateTrip createtrip;

            /**
             * @return the createtrip
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public CreateTrip getCreatetrip() {
                  // begin-user-code
                  return createtrip;
                  // end-user-code
            }

            /**
             * @param theCreatetrip the createtrip to set
```

```
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setCreatetrip(CreateTrip theCreatetrip) {
                  // begin-user-code
                  createtrip = theCreatetrip;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private Edittrip edittrip;

            /**
             * @return the edittrip
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Edittrip getEdittrip() {
                  // begin-user-code
                  return edittrip;
                  // end-user-code
            }

            /**
             * @param theEdittrip the edittrip to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setEdittrip(Edittrip theEdittrip) {
                  // begin-user-code
                  edittrip = theEdittrip;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void Enter_username() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void Enter_password() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }
```

```
            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void Forgot_p() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }
      }

      /**
       * @author Administrator
       * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
       */
      public class CreateTrip {
            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private Login_details login_details;

            /**
             * @return the login_details
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Login_details getLogin_details() {
                  // begin-user-code
                  return login_details;
                  // end-user-code
            }

            /**
             * @param theLogin_details the login_details to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setLogin_details(Login_details theLogin_details) {
                  // begin-user-code
                  login_details = theLogin_details;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private String Startpoint;

            /**
             * @return the Startpoint
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public String getStartpoint() {
                  // begin-user-code
```

```
                return Startpoint;
                // end-user-code
        }

        /**
         * @param theStartpoint the Startpoint to set
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public void setStartpoint(String theStartpoint) {
                // begin-user-code
                Startpoint = theStartpoint;
                // end-user-code
        }

        /**
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        private String Endpoint;

        /**
         * @return the Endpoint
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public String getEndpoint() {
                // begin-user-code
                return Endpoint;
                // end-user-code
        }

        /**
         * @param theEndpoint the Endpoint to set
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public void setEndpoint(String theEndpoint) {
                // begin-user-code
                Endpoint = theEndpoint;
                // end-user-code
        }

        /**
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        private Integer Depaturetime;

        /**
         * @return the Depaturetime
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public Integer getDepaturetime() {
                // begin-user-code
                return Depaturetime;
                // end-user-code
        }
```

```
            /**
             * @param theDepaturetime the Depaturetime to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setDepaturetime(Integer theDepaturetime) {
                  // begin-user-code
                  Depaturetime = theDepaturetime;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private Edittrip edittrip;

            /**
             * @return the edittrip
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Edittrip getEdittrip() {
                  // begin-user-code
                  return edittrip;
                  // end-user-code
            }

            /**
             * @param theEdittrip the edittrip to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setEdittrip(Edittrip theEdittrip) {
                  // begin-user-code
                  edittrip = theEdittrip;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private Edittrip edittrip2;

            /**
             * @return the edittrip2
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Edittrip getEdittrip2() {
                  // begin-user-code
                  return edittrip2;
                  // end-user-code
            }

            /**
             * @param theEdittrip2 the edittrip2 to set
```

```
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setEdittrip2(Edittrip theEdittrip2) {
                  // begin-user-code
                  edittrip2 = theEdittrip2;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void st_pt() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void end_pt() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void selectoptions() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void dept_time() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }
      }

      /**
       * @author Administrator
       * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
       */
```

```
      public class Edittrip extends Sharetrip {
            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private Login_details login_details;

            /**
             * @return the login_details
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Login_details getLogin_details() {
                  // begin-user-code
                  return login_details;
                  // end-user-code
            }

            /**
             * @param theLogin_details the login_details to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setLogin_details(Login_details theLogin_details) {
                  // begin-user-code
                  login_details = theLogin_details;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private String edittrip;

            /**
             * @return the edittrip
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public String getEdittrip() {
                  // begin-user-code
                  return edittrip;
                  // end-user-code
            }

            /**
             * @param theEdittrip the edittrip to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setEdittrip(String theEdittrip) {
                  // begin-user-code
                  edittrip = theEdittrip;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
```

```
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private String deletetrip;

            /**
             * @return the deletetrip
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public String getDeletetrip() {
                  // begin-user-code
                  return deletetrip;
                  // end-user-code
            }

            /**
             * @param theDeletetrip the deletetrip to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setDeletetrip(String theDeletetrip) {
                  // begin-user-code
                  deletetrip = theDeletetrip;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            private CreateTrip createtrip;

            /**
             * @return the createtrip
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public CreateTrip getCreatetrip() {
                  // begin-user-code
                  return createtrip;
                  // end-user-code
            }

            /**
             * @param theCreatetrip the createtrip to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setCreatetrip(CreateTrip theCreatetrip) {
                  // begin-user-code
                  createtrip = theCreatetrip;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void edit_st_pt() {
```

```
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void edit_end_pt() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void _new() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void deletetrip() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }
      }

      /**
       * @author Administrator
       * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
       */
      public class Sharetrip {
            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Object download;

            /**
             * @return the download
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public Object getDownload() {
                  // begin-user-code
```

```
                return download;
                // end-user-code
        }

        /**
         * @param theDownload the download to set
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public void setDownload(Object theDownload) {
                // begin-user-code
                download = theDownload;
                // end-user-code
        }

        /**
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public Object Share;

        /**
         * @return the Share
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public Object getShare() {
                // begin-user-code
                return Share;
                // end-user-code
        }

        /**
         * @param theShare the Share to set
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public void setShare(Object theShare) {
                // begin-user-code
                Share = theShare;
                // end-user-code
        }

        /**
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public Object save;

        /**
         * @return the save
         * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
         */
        public Object getSave() {
                // begin-user-code
                return save;
                // end-user-code
        }
```

```
            /**
             * @param theSave the save to set
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void setSave(Object theSave) {
                  // begin-user-code
                  save = theSave;
                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void Download() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void Share() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }

            /**
             * @generated "UML to Java V5.0
(com.ibm.xtools.transform.uml2.java5.internal.UML2JavaTransform)"
             */
            public void Save() {
                  // begin-user-code
                  // TODO Auto-generated method stub

                  // end-user-code
            }
      }
}
```

Configuration Management

AIM:

To Perform a) Configuration Management

HARDWARE REQUIREMENTS:

Pentium III, 256MBRAM, 80GB HDD, keyboard, mouse, monitor.

SOFTWARE REQUIREMENTS: Windows XP, IBM Quality Manager

DESCRIPTION:

Software Configuration Management

Current definition would say that SCM is the control of the evolution of complex systems. More pragmatically, it is the discipline that enables us to keep evolving software products under control, and thus contributes to satisfying quality and delay constraints.

The procedure for managing the test object and related test ware should be described.

The version Management of the test ware is ultimately the test team responsibility.

One of the issues that must be addressed in Configuration is that modified objects may only be installed in the test environment with the test team permission, after action has been taken on the basis of the errors. This is to prevent a test from failing when a different version of the object under test is unexpectedly being used.

Rational Quality Manager

Test Plan : DISPLAY STOPS

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : Showing the display stops such as petrol bunks, restaurants etc.

Rational Quality Manager

Test Plan : TRIP DETAILS

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : Printing trip details after the trip has completed.

Rational Quality Manager

Test Plan : LOGIN

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : ALLOW THE THE USERS TO LOGIN THROUGH THEIR IDS.

Requirements

Status	ID	Name	Description	Owner
New	30	LOGIN	ALLOW THE THE USERS TO LOGIN THROUGH THEIR IDS.	unassigned

Rational Quality Manager

Test Plan : EDIT TRIP

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : HELPS THE USERS TO EDIT THE TRIP DETAILS.

Requirements

Status	ID	Name	Description	Owner
New	33	EDIT TRIP	HELPS THE USERS TO EDIT THE TRIP DETAILS.	unassigned

Rational Quality Manager

Test Plan : CREATE TRIP

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : HELPS ONE TO ACCESS TO REQUIRED DETAILS OF THE TRIP.

Requirements

Status	ID	Name	Description	Owner
New	31	CREATE TRIP	HELPS ONE TO ACCESS TO REQUIRED DETAILS OF THE TRIP.	unassigned

Rational Quality Manager

Test Plan : SHARE TRIP

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Un

assigned

Description : THE USERS GET TO SHARE THE TRIP OR DOWNLOAD THE TRIP. **Requirements**

Status	ID	Name	Description	Owner
New	35	SHARE TRIP	THE USERS GET TO SHARE THE TRIP OR DOWNLOAD THE TRIP.	unassigned

Rational Quality Manager

https://localhost:9443/jazz/web/console/Active%20Park%20Assist#action=com.ibm.rqm.planning.home.actionDispatcher&subAction=viewTestPlan&id=7

Test Plan : SAVE TRIP

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : HELPS THE USERS TO SAVE THE TRIP DETAILS.

Requirements

Status	ID	Name	Description	Owner
New	34	SAVE TRIP	HELPS THE USERS TO SAVE THE TRIP DETAILS.	unassigned

Rational Quality Manager

https://localhost:9443/jazz/web/console/Active%20Park%20Assist#action=com.ibm.rqm.planning.home.actionDispatcher&subAction=viewTestPlan&id=7

Test Plan : SELECT ROUTE

Originator : AKSHITHA

State : Draft

Summary

Product : Unassigned
Release : Unassigned

Description : HELP USERS TO SELECT TO SELECT START AND END LOCATIONS VIA TRIP.

Requirements

Status	ID	Name	Description	Owner
New	32	SELECT ROUTE	HELP USERS TO SELECT TO SELECT START AND END LOCATIONS VIA TRIP.	unassigned

Rational Quality Manager

Summary : Test Plan

ID	Name	State	Product	Release	Modified Date
2	LOGIN	Draft	Unassigned	Unassigned	6/8/22 4:04 PM
3	CREATE TRIP	Draft	Unassigned	Unassigned	6/8/22 4:05 PM
4	EDIT TRIP	Draft	Unassigned	Unassigned	6/8/22 4:05 PM
5	SHARE TRIP	Draft	Unassigned	Unassigned	6/8/22 4:05 PM
6	SAVE TRIP	Draft	Unassigned	Unassigned	6/8/22 4:06 PM
7	TRIP DETAILS	Draft	Unassigned	Unassigned	6/8/22 4:1 1 PM
8	DISPLAY STOPS	Draft	Unassigned	Unassigned	6/8/22 4:1 1 PM

Rational Quality Manager

Test Case : EDITING THE DETAILS OF TRIP

Originator : AKSHITHA

State : Draft

Summary

Category : Unassigned
Function : Unassigned
Theme : Unassigned
Test Activity : Unassigned
Trigger : Unassigned
Weight : 100

Description : HELPS THE USER TO EDIT THE DETAILS.

Requirements

Status	ID	Name	Description	Owner
New	33	EDIT TRIP	HELPS THE USER TO EDIT THE DETAILS.	SHRUTHI

Work Items

Type	ID	Summary	State
Task-Quality	58	Provide the RQM-KEY-TC-SUMMARY-TITLE Section for TestCase: EDITING DETAILS OF THE TRIP	New

Rational Quality Manager

Test Case : INSERT DETAILS OF TRIP

Originator : AKSHITHA

State : Draft

Summary

Category : Unassigned
Function : Unassigned
Theme : Unassigned
Test Activity : Unassigned
Trigger : Unassigned
Weight : 100

Description : HELPS ONE TO ACCESS THE DETAILS OF THE TRIP.

Requirements

Status	ID	Name	Description	Owner
New	31	CREATE TRIP	HELPS ONE TO ACCESS THE DETAILS OF THE TRIP.	SHRUTHI

Work Items

Type	ID	Summary	State
Task-Quality	54	Provide the RQM-KEY-TC-SUMMARY-TITLE Section for TestCase: INSERT THE DETAILS OF THE TRIP	New

Rational Quality Manager

Test Case : Login

Originator : AKSHITHA

State : Draft

Summary

Category : Unassigned
Function : Unassigned
Theme : Unassigned
Test Activity : Unassigned
Trigger : Unassigned
Weight : 100

Description : ALLOW THE USERS TO LOGIN THROUGH THEIR IDS.

Requirements

Status	ID	Name	Description	Owner
New	30	Login	ALLOW THE USERS TO LOGIN THROUGH THEIR IDS.	SHRUTHI

Work Items

Type	ID	Summary	State
Task-Quality	51	Provide the RQM-KEY-TC-SUMMARY-TITLE Section for TestCase: Login	New

Rational Quality Manager

Summary: Requirements

Status	ID	Name	Description	Owner
New	32	SELECT ROUTE	HELPS USERS TO SELECT START AND END LOCATIONS VIA TRIP.	unassigned
New	34	SAVE TRIP	HELPS THE USERS TO SAVE THE TRIP DETAILS.	unassigned
New	35	SHARE TRIP	THE USERS GET TO SHARE THE TRIP OR DOWNLOAD THE TRIP.	unassigned
New	31	CRATE TRIP	Helps one to access to required details of the trip.	unassigned
New	33	EDIT TRIP	HELPS THE USERS TO EDIT THE TRIP DETAILS.	unassigned
New	30	LOGIN	Allow the the users to login through their ids.	unassigned

8 June, 2022 4:35:12 PM IST Script start [CODIAC]
• line_number = 1 • script_iter_count = 0 • script_name = CODIAC • script_id = CODIAC.java
8 June, 2022 4:35:12 PM IST Start timer: ClassicsCD_1
• simplifiedscript_group_name = [ClassicsCD] • name = ClassicsCD_1 • simplifiedscript_line_number = 1 • simplifiedscript_group_name = [ClassicsCD] • line_number = 44 • script_name = CODIAC • script_id = CODIAC.java
8 June, 2022 4:35:13 PM IST Stop timer: ClassicsCD_1
• simplifiedscript_group_name = [ClassicsCD] • name = ClassicsCD_1 • simplifiedscript_line_number = 2 • simplifiedscript_group_name = [ClassicsCD] • line_number = 48 • script_name = CODIAC • script_id = CODIAC.java • additional_info = Elapsed time: 0.198 secs. • elapsed_time = Elapsed time: 0.198 secs.
8 June, 2022 4:35:13 PM IST Start timer: MemberLogon_3
• simplifiedscript_group_name = [Member Logon] • name = MemberLogon_3 • simplifiedscript_line_number = 3 • simplifiedscript_group_name = [Member Logon] • line_number = 51 • script_name = CODIAC • script_id = CODIAC.java
8 June, 2022 4:35:13 PM IST Stop timer: MemberLogon_3
• simplifiedscript_group_name = [Member Logon] • name = MemberLogon_3 • simplifiedscript_line_number = 5 • simplifiedscript_group_name = [Member Logon] • line_number = 58 • script_name = CODIAC • script_id = CODIAC.java • additional_info = Elapsed time: 0.205 secs. • elapsed_time = Elapsed time: 0.205 secs.
8 June, 2022 4:35:13 PM IST Start timer: PlaceanOrder_6

- simplifiedscript_group_name = [Place an Order]
- name = PlaceanOrder_6
- simplifiedscript_line_number = 6
- simplifiedscript_group_name = [Place an Order] • line_number = 61

- script_name = CODIAC
- script_id = CODIAC.java

8 June, 2022 4:35:15 PM IST_Stop timer: PlaceanOrder_6

- simplifiedscript_group_name = [Place an Order]
- name = PlaceanOrder_6
- simplifiedscript_line_number = 24
- simplifiedscript_group_name = [Place an Order]
- line_number = 116
- script_name = CODIAC
- script_id = CODIAC.java
- additional_info = Elapsed time: 2.669 secs.
- elapsed_time = Elapsed time: 2.669 secs.

8 June, 2022 4:35:15 PM IST_Start timer: Message_25

- simplifiedscript_group_name = [Message]
- name = Message_25
- simplifiedscript_line_number = 25
- simplifiedscript_group_name = [Message]
- line_number = 119
- script_name = CODIAC
- script_id = CODIAC.java

8 June, 2022 4:35:16 PM IST_Stop timer: Message_25

- simplifiedscript_group_name = [Message]
- name = Message_25
- simplifiedscript_line_number = 26
- simplifiedscript_group_name = [Message]
- line_number = 123
- script_name = CODIAC
- script_id = CODIAC.java
- additional_info = Elapsed time: 0.082 secs.
- elapsed_time = Elapsed time: 0.082 secs.

8 June, 2022 4:35:16 PM IST_Start timer: ClassicsCD_27

- simplifiedscript_group_name = [ClassicsCD]
- name = ClassicsCD_27
- simplifiedscript_line_number = 27
- simplifiedscript_group_name = [ClassicsCD]
- line_number = 126
- script_name = CODIAC
- script_id = CODIAC.java

8 June, 2022 4:35:16 PM IST_Stop timer: ClassicsCD_27

- simplifiedscript_group_name = [ClassicsCD]
- name = ClassicsCD_27
- simplifiedscript_line_number = 28
- simplifiedscript_group_name = [ClassicsCD]
- line_number = 130
- script_name = CODIAC
- script_id = CODIAC.java
- additional_info = Elapsed time: 0.095 secs.
- elapsed_time = Elapsed time: 0.095 secs.

8 June, 2022 4:35:16 PM IST **Start timer: MemberLogon_29**

- *simplifiedscript_group_name* = [Member Logon]
- *name* = MemberLogon_29
- *simplifiedscript_line_number* = 29
- *simplifiedscript_group_name* = [Member Logon]
- *line_number* = 133
- *script_name* = CODIAC
- *script_id* = CODIAC.java

8 June, 2022 4:35:17 PM IST **Stop timer: MemberLogon_29**

- *simplifiedscript_group_name* = [Member Logon]
- *name* = MemberLogon_29
- *simplifiedscript_line_number* = 30
- *simplifiedscript_group_name* = [Member Logon]
- *line_number* = 137
- *script_name* = CODIAC
- *script_id* = CODIAC.java
- *additional_info* = Elapsed time: 1.121 secs.
- *elapsed_time* = Elapsed time: 1.121 secs.

8 June, 2022 4:35:17 PM IST **Start timer: PlaceanOrder_31**

- *simplifiedscript_group_name* = [Place an Order]
- *name* = PlaceanOrder_31
- *simplifiedscript_line_number* = 31
- *simplifiedscript_group_name* = [Place an Order]
- *line_number* = 140
- *script_name* = CODIAC
- *script_id* = CODIAC.java

8 June, 2022 4:35:17 PM IST **Stop timer: PlaceanOrder_31**

- *simplifiedscript_group_name* = [Place an Order]
- *name* = PlaceanOrder_31
- *simplifiedscript_line_number* = 38
- *simplifiedscript_group_name* = [Place an Order]
- *line_number* = 162
- *script_name* = CODIAC
- *script_id* = CODIAC.java
- *additional_info* = Elapsed time: 0.622 secs.
- *elapsed_time* = Elapsed time: 0.622 secs.

8 June, 2022 4:35:17 PM IST **Start timer: IncompleteOrder_39**

- *simplifiedscript_group_name* = [Incomplete Order]
- *name* = IncompleteOrder_39
- *simplifiedscript_line_number* = 39
- *simplifiedscript_group_name* = [Incomplete Order]
- *line_number* = 165
- *script_name* = CODIAC
- *script_id* = CODIAC.java

8 June, 2022 4:35:18 PM IST **Stop timer: IncompleteOrder_39**

- *simplifiedscript_group_name* = [Incomplete Order]
- *name* = IncompleteOrder_39
- *simplifiedscript_line_number* = 40

- simplifiedscript_group_name = [Incomplete Order]
- line_number = 169
- script_name = CODIAC
- script_id = CODIAC.java
- additional_info = Elapsed time: 1.081 secs.
- elapsed_time = Elapsed time: 1.081 secs.

8 June, 2022 4:35:18 PM IST Start timer: PlaceanOrder_41

- simplifiedscript_group_name = [Place an Order]
- name = PlaceanOrder_41
- simplifiedscript_line_number = 41
- simplifiedscript_group_name = [Place an Order]
- line_number = 172
- script_name = CODIAC
- script_id = CODIAC.java

8 June, 2022 4:35:19 PM IST Stop timer: PlaceanOrder_41

- simplifiedscript_group_name = [Place an Order]
- name = PlaceanOrder_41
- simplifiedscript_line_number = 47
- simplifiedscript_group_name = [Place an Order]
- line_number = 191
- script_name = CODIAC
- script_id = CODIAC.java
- additional_info = Elapsed time: 0.828 secs.
- elapsed_time = Elapsed time: 0.828 secs.

8 June, 2022 4:35:19 PM IST Start timer: Message_48

- simplifiedscript_group_name = [Message]
- name = Message_48
- simplifiedscript_line_number = 48
- simplifiedscript_group_name = [Message]
- line_number = 194
- script_name = CODIAC
- script_id = CODIAC.java

8 June, 2022 4:35:20 PM IST Stop timer: Message_48

- simplifiedscript_group_name = [Message]
- name = Message_48
- simplifiedscript_line_number = 49
- simplifiedscript_group_name = [Message]
- line_number = 198
- script_name = CODIAC
- script_id = CODIAC.java
- additional_info = Elapsed time: 1.078 secs.
- elapsed_time = Elapsed time: 1.078 secs.

PASS 8 June, 2022 4:35:20 PM IST Script end [CODIAC]

- simplifiedscript_group_name = [Message]
- script_name = CODIAC
- script_id = CODIAC.java

Conclusion:

The process of Road Trip Planner has been fully automated with this software. This web app can now avail to every user and can be used easily to plan a road trip. Now we don't need to worry about our trip anymore.

GitHub : https://github.com/AkshithaDoodala/ROAD-TRIIP-PLANNER

References:

https://www.tutorialspoint.com/uml/uml_standard_diagrams.htm
https://www.geeksforgeeks.org/software-engineering-system-configuration-management/
https://www.javatpoint.com/functional-testing