Table of Contents

Software Requirements Specifications (SRS)

1. Introduction

1.1 Purpose of this Document

This SRS document describes the functions and requirements for the GPS Tracking System.

1.2 Scope of the Development Project

The product that will be developed is a GPS Tracking System. This product uses both hardware, software, and many technologies to support the task at hand. This product will be able to have add-ons specific to the needs of various industries. We will not implement add-ons for all the different possible users of this product. We will however implement add-ons for one or two industries like the car rental agency or a trucking company

1.3 Definitions, Acronyms, and Abbreviations

GPS: Global Positioning System is a worldwide radio-navigation system formed from a constellation of 24 satellites and their ground stations.

GPS Receiver: Electronic device that creates data strings based on satellite positions that contains data on: latitude, longitude, speed, heading, signal strength, and many other pieces of information.

Longitude: The angular distance east or west of the earth's equator, measured in degrees along a meridian, as on a map or globe.

Latitude: The angular distance north or south of the earth's equator, measured in degrees along a meridian, as on a map or globe.

Web Client: It is a computer interface that utilizes a Web server usually through a web browser. Our web client will connect to our tracking information web server

page. This page will contain information on every GPS coordinate for the specified login. This client will also have mapping where a user can visually see their vehicles move on the map.

Vehicle Client: This is the software that will be on the computer in the vehicle. This application will be in charge of communicating with the GPS receiver, parsing the data and then sending it to our server.

Carputer: This is a computer, usually in a smaller form factor from a desktop client that has a special power supply for starting up and shutting down with the car.

Database: A collection of information stored in a computer in a systematic way. This will refer to a place on our server.

SQL: Structured Query Language (SQL), pronounced "sequel", is a language that provides an interface to relational database systems.

.Net Connectors: API's used for communicating with different databases written for .NET languages.

Web Server: A computer including software package, which provides a specific kind of service to client software running on other computers. Our Web Server will receive data requests from all our different client types.

AJAX: Asynchronous JavaScript and XML is a term describing a web development technique for creating interactive web applications

Google Maps: A free, online map service provided by Google at http://maps.google.com.

2. General Description

2.1 User Personas and Characteristics

Our product will potentially have many users, as there may be many add-ons to the product each one catering to different types of users in various industries.

- Home User – A user who wants to use the product to keep track of his vehicles at home. They won't need to have extended functionality. See company employee section
- Company Employee – This is a user who uses the product to make his life easier and to perform specific tasks. Such tasks for a Rental Agency, for example, would be to keep track of all the vehicles in the arsenal and see if any of the vehicles goes out of a specific geographic area.
- Company Manager – This user will need the ability to add more vehicles in his arsenal of vehicles to manage, or to remove a vehicle from the arsenal.

2.2 Product Perspective

- The web server will run on ASP
- The Pocket PC portion will be run on the .Net Compact Framework
- GPS Hardware must be connected to a computer to collect the GPS coordinates.
- The Vehicle Client will need an internet connection to upload current GPS coordinates.
- The product will need a Carputer which is assumed the client will already have installed.
- The product will utilize AJAX and Google Maps
- The product will have limited use on a PDA that the user must supply
- The Desktop Client must have the Compact Framework installed on their windows machine.

2.3 Overview of Functional Requirements

1. Input GPS Coordinates in Database in a structured format.
2. View GPS locations using AJAX & Google Maps
3. Look-up vehicle numbers and retrieve information such as last known coordinates, average speed, etc.
4. User can login and add vehicles, remove vehicles, and see vehicles associated with their account
5. View statistics: average speed for all vehicles, distance traveled, locations, etc.
6. Multiple interfaces to interact with the data (Web, Client, PDA)
7. Provide different levels of access based on the user's security (restricted viewing, administrative, etc.)

2.4 Overview of Data Requirements

Users will be able to set up a login name and password to access our system. Interfaces to our system will include a Web version, Client version and PDA version. They will be able to store information about cars in the database by means of a vehicle number (license plate and state). They will be able to check the location, speed, heading, vehicle history etc. of a car and view this information on visual maps provided by Google Maps.

2.5 General Constraints, Assumptions, Dependencies, Guidelines

1. The product will only work with Windows 98 and up.
2. The product will only work with current releases of web browsers.
3. The PDA will not have full functionality and limited storage space.
4. Users will be responsible for functioning GPS hardware, wireless connection, etc. in their car(s).
5. This product will the .Net framework.
6. For the desktop Client to work there must be open ports on the computer. Firewalls must be set with open ports.

2.6 User View of Product Use

Scenario 1 (the vehicle client):

This scenario occurs when a user turns on their vehicle and then turns on their car computer, or waits for their car computer to turn on its self:

Vehicle client starts with Windows. Excluding the initial setup of the vehicle client there should be no need for human interaction. The application will run in the

system tray and upload its positional information to the server, anywhere from 15 seconds to 5 minutes apart. If this is an employee of a company that is using our software then they will possibly need to log in with their vehicle ID and/or their employee ID. There will be a mini-mode option that will consist of a small square above the system clock showing the status of the GPS signal, position logging, and log upload. The user may double click on this to bring up a larger status window showing more information including latitude/longitude, speed, direction and satellite positional information.

Scenario 2 (the web client):

This scenario occurs when either a consumer or a company employee, using our software, logs into our tracking page using their login information:

The user will log into the tracking page. If this is a consumer's login then they will be presented with tracking information for their own vehicle. The user will be able to select from a calendar the day from which they wish to see their positional logs. Upon selecting the day a list box will be filled with GPS fix times. The user will be able to select a time and see positional information like: latitude/longitude, speed, direction, and distance. The user will also be able to see their position on a map for a specific time or see a marker representing their vehicle move along on a map showing their position throughout a day.

If the login information is for a company employee using our product they will be presented with a page that will allow them to select from specific vehicles and different tickets. Each vehicle page will be similar to the consumer vehicle page. Inside the ticket section of the web client the user will be able to view and edit ticket information for dynamic tickets set up by the company using our product. They will be able to select what type of information will be used in their tickets (i.e. customer address, ticket number, product, cost etc...) depending on what industry they are in.

Scenario 3 (the web server):

There will be no customer interaction with this part of our product.

Scenario 4 (the desktop client):

This will act the same way as the Web Client except as a Windows application.

3. Specific Requirements

3.1 External Interface Requirements

- User application will be run using any up to date Web Browser.
- The GPS System will communicate with the computer in the car which will upload the coordinates to the web server.
- The Pocket PC application will communicate with the web server through the WIFI capabilities on the Pocket PC device.

3.2 Performance Requirements

Each vehicle will only have one web client. The Web Server will be able to handle any number of vehicles updating to the server, and any number of users logged in. The Database will need to support a lot of different GPS coordinates and keep track of what coordinates go with what vehicle.

3.3 Quality Attributes

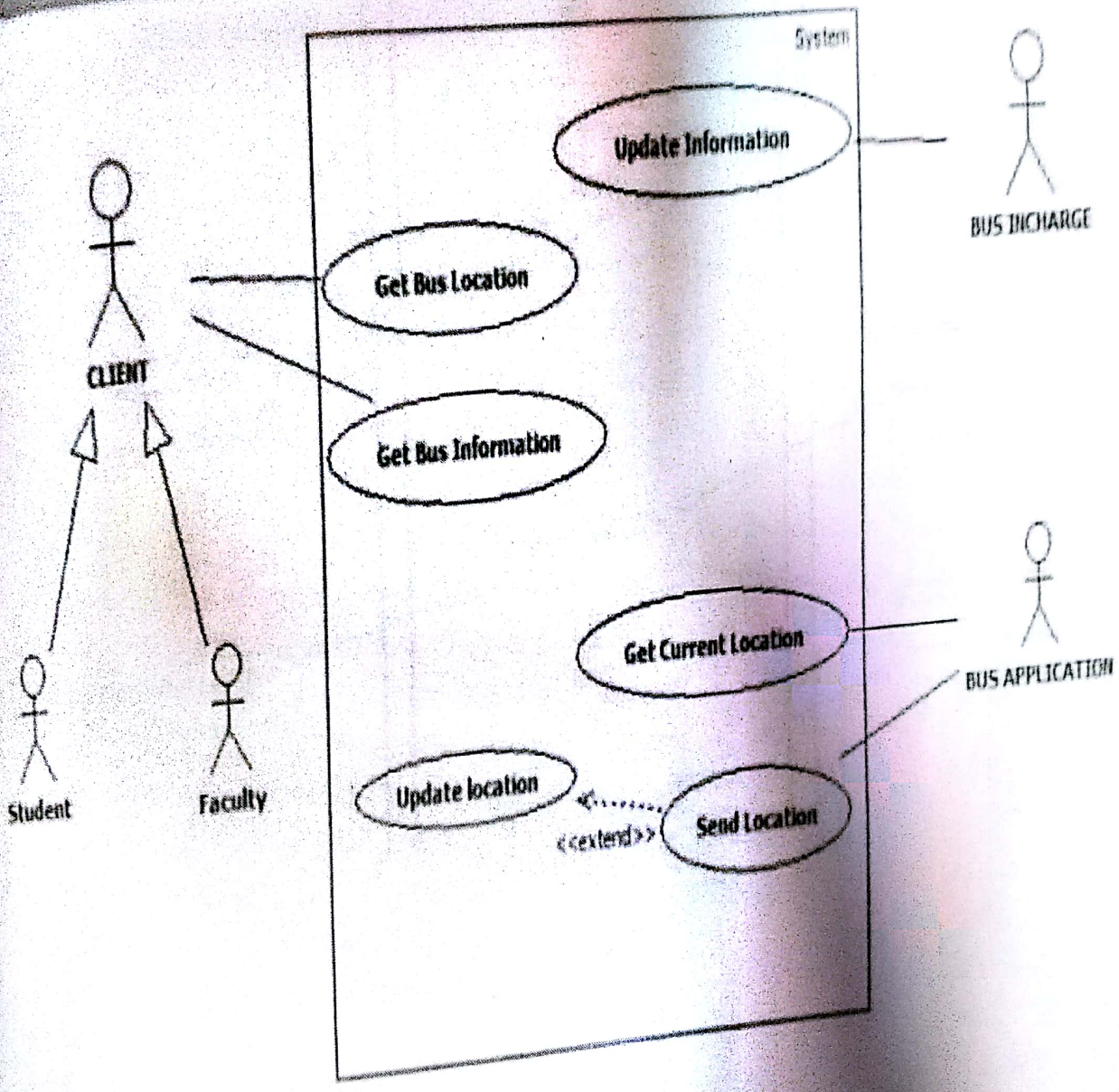
A history of known issues will be kept as the project goes on. We will keep track of when the item is fixed, and all the issues will have a current status. For the product to be finished, 90% of the known issues will be fixed. All the high priority items will also be fixed. This will allow for a stable product. We will perform unit tests on the different modules before integrating. A history of the unit tests will also be kept. Each of these documents will be available for edit by team members, and by none else.

Security to the Web App will be of utmost importance. We will verify that users without a valid matching user name and password will not be able to access and change other users' information.

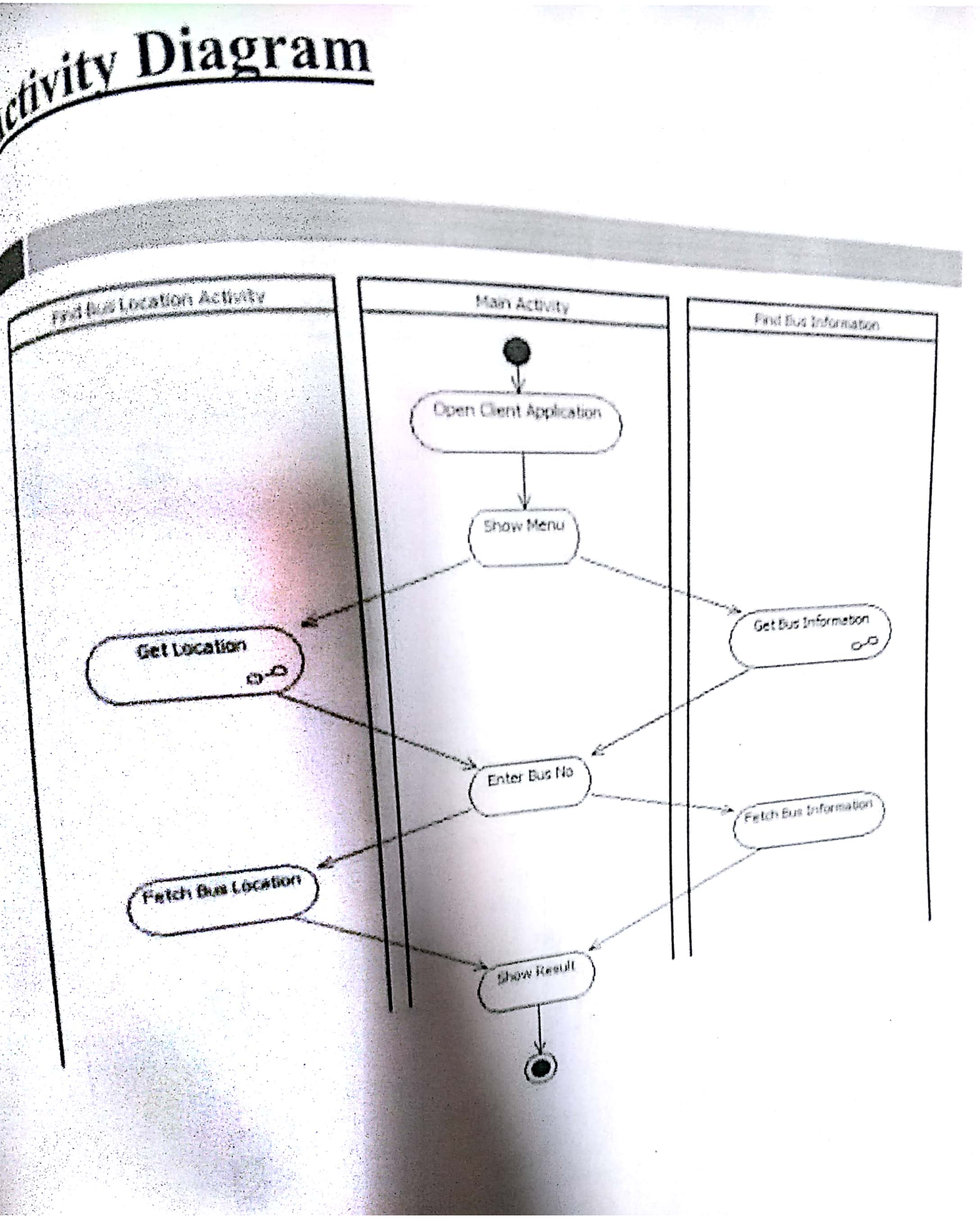
3.4 Other requirements

None at this time.

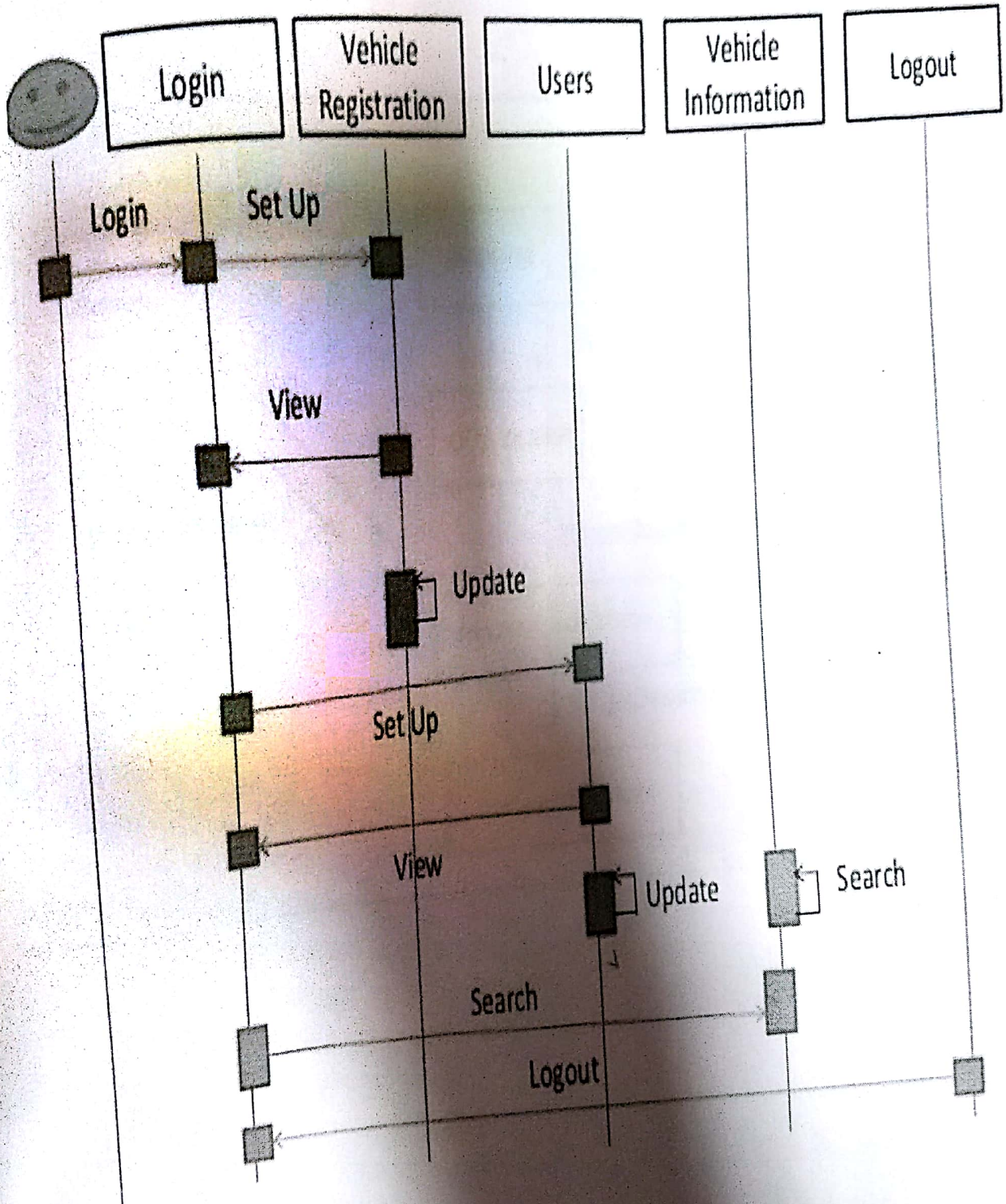
se Case Diagram



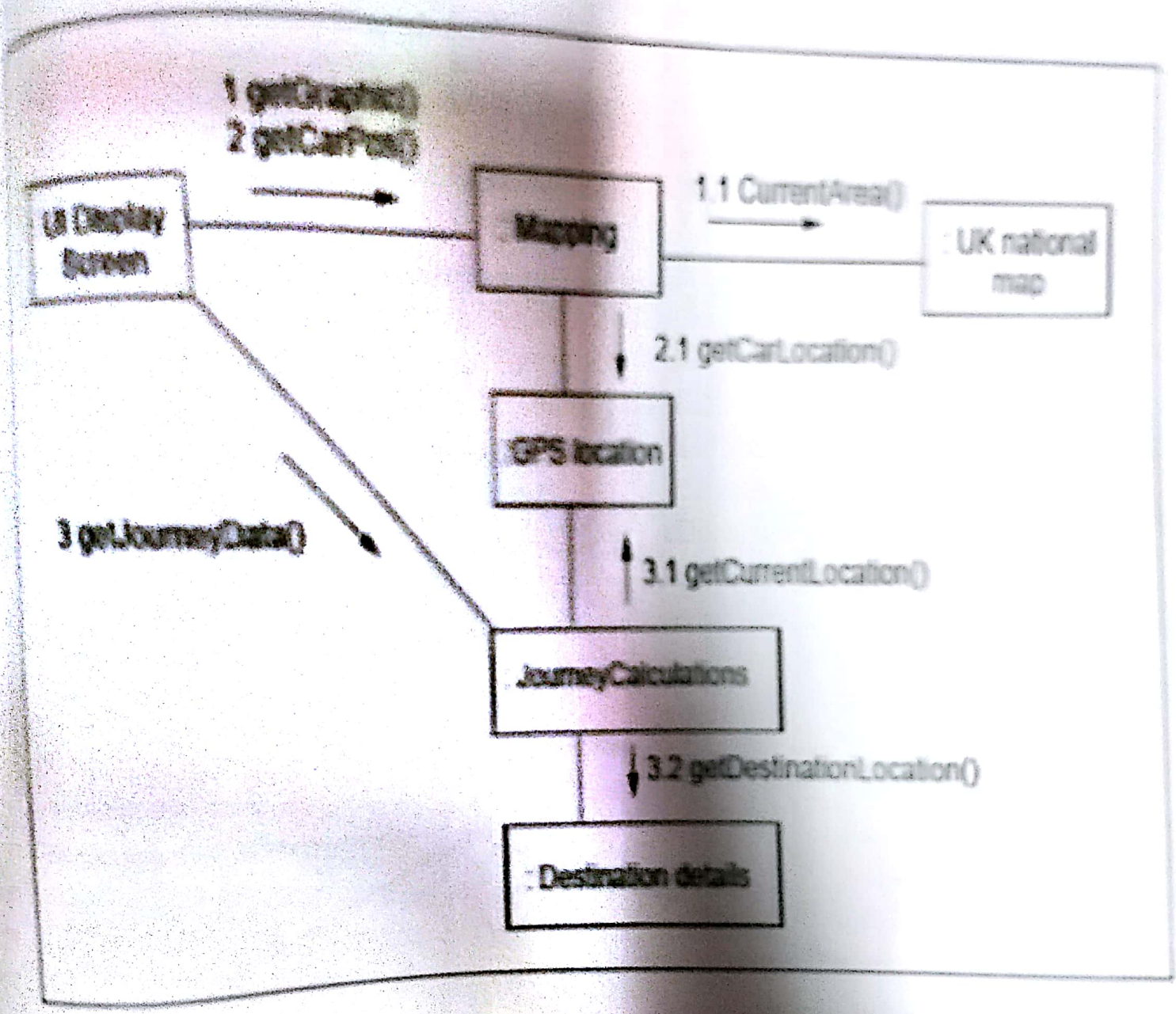
ctivity Diagram



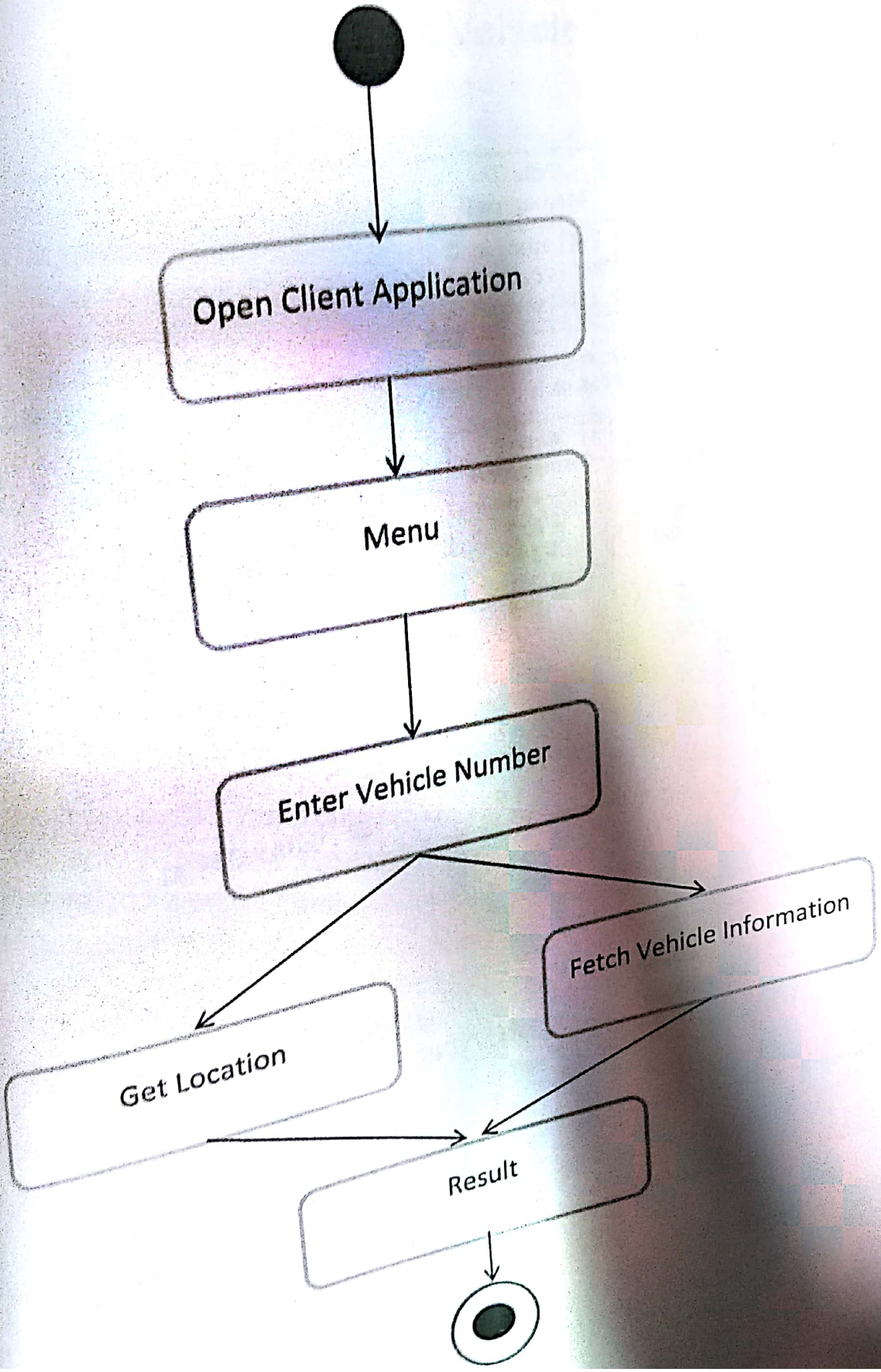
Sequence Diagram



llaboration Diagram

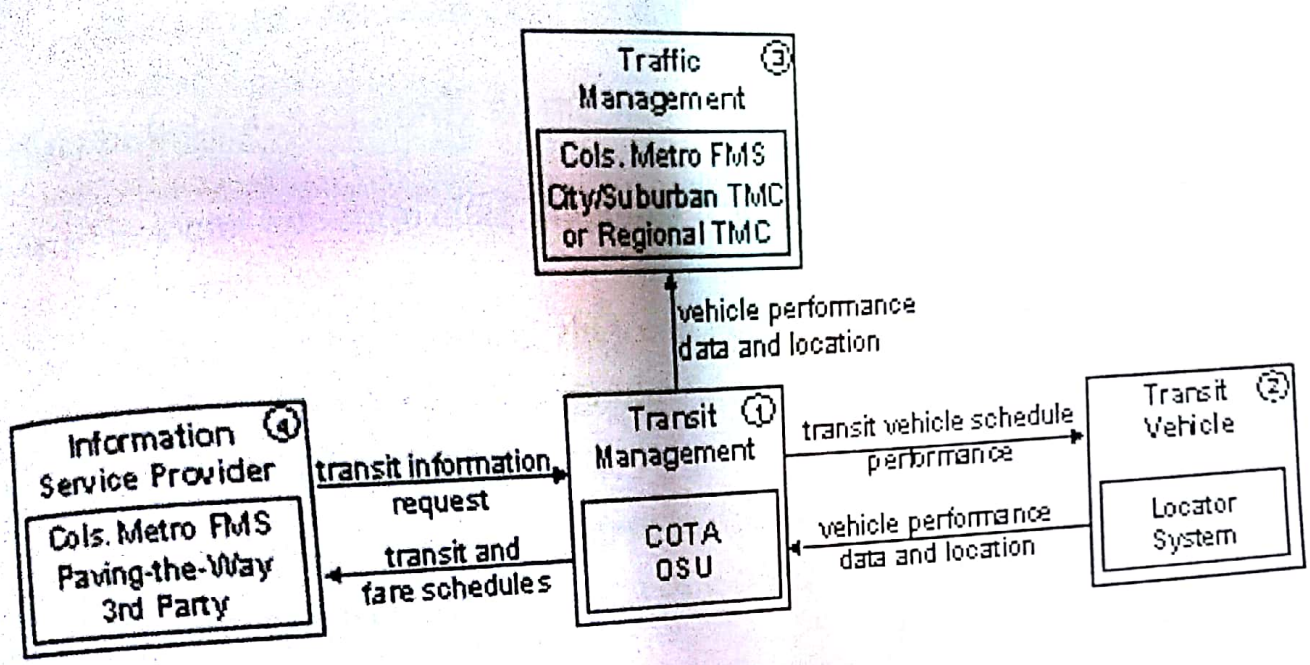


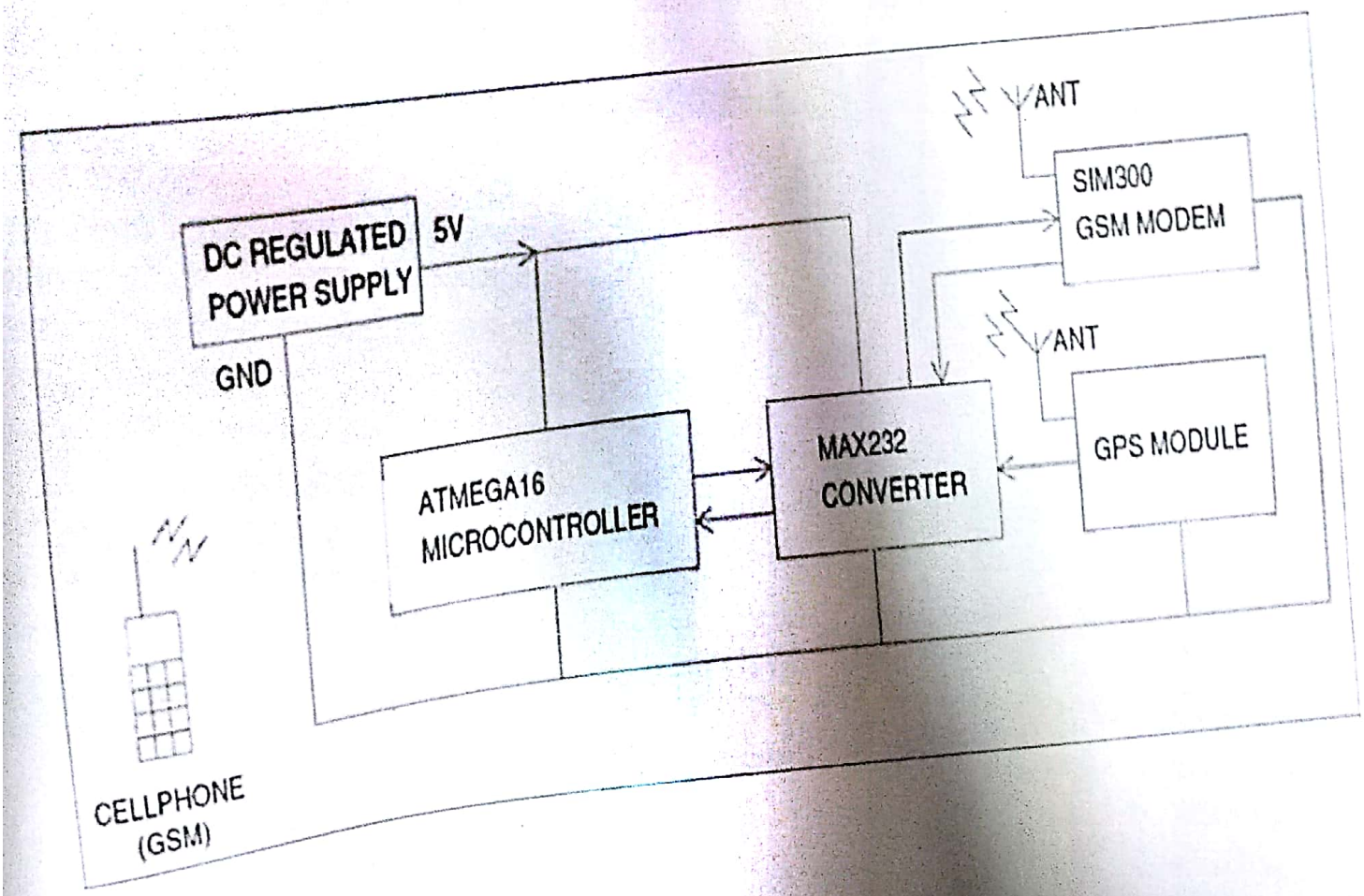
State Diagram



Component Diagram

Transit Vehicle Tracking





orward Engineering

Forward engineering means the generation of code from UML diagrams. The forward engineering process applies software engineering principles, concepts and methods to re-create an existing application.

Many of the tools can only do the static models:

- They can generate class diagrams from code, but can't generate interaction diagrams.
- For forward engineering, they can generate the basic (e.g., Java) class definition from a class diagram, but not the method bodies from interaction diagrams.

Forward engineering is an attempt to develop physical model of a management system from its logical model.