BATTLEFIELD INFORMATION AND MANAGEMENT SYSTEM

By

Atif Khan Jadoon
Waqas Ahmed Choudhary
Ahmed Abdul Rehman

Submitted to the Faculty of Computer Software Engineering Department, National University of Sciences and Technology, Islamabad, in partial fulfillment for the requirements of a B.E. Degree in Computer Software Engineering

June 2013

CERTIFICATE

Certified that the contents and form of project report entitled "**Battlefield Information and Management System**" submitted by 1) Atif Khan Jadoon, 2) Waqas Ahmed Choudhary, and 3) Ahmed Abdul Rehman under the supervision of Dr Adnan Rashdi for partial fulfillment of Degree of Computer Software Engineering, have been found satisfactory.

Supervisor: ______________________________

Dr Adnan Rashdi

ABSTRACT

Operational intelligence and flow of information during a military operation plays a vital role in its success. Contemporary armies have developed such systems that can apprise the higher commanders about the on ground operational scenario and can take orders for further execution of the plan.

Today, all the leading Main Battle Tanks of the world are equipped with same technology e.g., M1A1 Abraham, Leopard, Leclerc. Pakistan Army is currently working on such system named “Integrated Battlefield Management System”.

The thesis presents our project “Battlefield Information & Management System” (BIMS), which is aimed at developing a system to transfer data packets over the radio communication currently in use. Main objective of BIMS is to update the present location of moving echelons, and pass mission-critical information related to mechanized columns to the higher commanders.

The essence of all the information in exchange is the use of communication modules in the latest Software Defined Radios currently being inducted in Pakistan Army. Since, these radio sets are going to replace the older ones, and will be used as a standard for communication, their utilization as medium for transferring data packets is of prime importance & this is the primary objective of our project.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the Name of Allah, The Most Beneficent, The Most Merciful

This project thesis is dedicated to

Parents, Teachers

&

SowarSarfarazShaheed – 52 C
Sowar Shoaib Shaheed – 52 C
OCU Matloob Hussain Shaheed – 12 Med

who left us fighting in Operation Rah-e-Nijat (2009) and Operation Sunrise (2008).

ACKNOWLEDGEMENTS

"All praises be to Allah Almighty, The Most Exalted and The Most Dignified, Who guides us from the depths of darkness into the light and help us in difficult times"

We are grateful and thankful to Allah Almighty for His unremitting blessings, bestowed upon us throughout our project. Secondly, we are thankful to our families for their unending support that they have shown in good and hard times. Their perseverance andobstinate support helped us achieve much in the project.

We are greatly obliged to our project supervisor Dr Adnan Rashdi, who led us from the start, encouraged us in the times of difficulties, and provided us with utmost technical knowledge about the project. He has been very kind throughout our project and has played pivotal role in making us complete it. He has been there for us whenever we needed him. We again thank him for his support and cooperation, without which this project would not have been possible.

We are also indebted to our classmates and friends, who provided every possible support that they could, and helped us achieve technical expertise and move further through difficult areas of the project. Their selfless support contributed much to our project in the form of discussions, ideas exchange, and as a source of encouragement.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1: Introduction

1.1. Introduction

Operational intelligence and flow of information during a military operation plays a vital role in its success. Contemporary armies have developed such systems that can apprise the higher commanders about the on ground operational scenario and can take orders for further execution of the plan. Today, all the leading Main Battle Tanks of the world are equipped with same technology e.g., M1A1 Abraham, Leopard, Leclerc. Pakistan Army is currently working on such system named "Integrated Battlefield Management System".

The project "Battlefield Information & Management System" (BIMS) is aimed at developing a system to transfer data packets over the radio communication currently in use. Main objective of BIMS is to update the present location of moving echelons, and pass mission-critical information related to mechanized columns to the higher commanders.

1.2. Background

The advancement of technology in the field of computer sciences has led to tremendous breakthroughs with its applications. Contemporary armies today have put this advancement to their use by employing latest trends in communications and software development. One of the most famous example is of Blue Force Tracking, which has become the standard being used by all the assets of United States Army.

All these developments gave birth to Network Centric Warfare, which is a precursor to a computerized battlefield, where a high ranking commander can monitor and order his troops with a click. Keeping in mind the scenario, efforts of our team were focused on development of a system that supports communication via latest Software Defined Radios (SDRs) used by our army, update and track the troop, logistic and administrative information, thus creating a situational awareness of the operation of theatre in battle.

Battlefield Information and Management System is a system that manages and keeps records of all the activities in a battlefield. This also includes the Contact Reports, which are generated when a contact is being made with the enemy, subsequently enabling the high commander and the commander on ground to take actions against such threats in a coordinated manner. Therefore, the main essence of this project is to help coordinate the efforts of all the elements, human and equipment, participating in an operation or a battle.

1.3. Problems Addressed

The need for developing such system has a strong footing based on the following problems, existing in present systems, which are being addressed:

a. **Information Flow.** It is restricted to Radio Frequency (RF) over voice channel.

b. **Difficulty in Acquiring Troops' Location.**Acquiring own location is totally dependent on handheld GPS whereas the under command troops use RF communication channels for location passing.

c. **Difficulty in Closed Down.** Lack of visual contact while closing down makes the adjacent troops' location unknown to the commander.

d. **Lack of Operational Information at Higher Level(s).**No update of dynamic information of the battlefield, or the Area of Operation, is available with the higher echelons of command.

1.4. Goals and Objectives

The Battlefield Information and Management System emphasized on achieving under mentioned goals and objective, both technical and academic:

a. **Goal.** A system for network packet creating / crafting, its delivery to the desired location and provide services based on the communicated information.

b. **Scope.** A system to create, manage and communicate information related to mechanized columns and moving echelons.

c. **Objectives.** The project / system will entail the following objectives:

(1) **Technical.** Following areas are covered:
- (a) Situation Awareness.
- (b) AFV Troops' Management.
- (c) Command and Control Console.
- (d) Geographical Information System.
- (e) Packet Manager.

(2) **Academic.** Following technologies are used:
- (a) Microsoft Visual Studio 2012.
- (b) Windows CE (for tablets of Windows)
- (c) Google APIs (for Maps)
- (d) Android SDK (for tablets with Android)
- (e) PHP & MySQL (for database(s) & interface)

1.5. Deliverables

Phase wise deliverables during the project development are as under:

a. **Phase I.**
- (1) Project Proposal & Synopsis.
- (2) Requirements Analysis Report.
- (3) Software Requirement Specifications
- (4) First Progress Report.

b. **Phase II.**
- (1) 2nd Progress Report.
- (2) Detailed Design Document.

c. **Phase III.**
- (1) Development and Implementation scheme.
- (2) Development of modules separately.
- (3) Software prototype produced.
- (4) Software installation & configurations with customizations.

d. **Phase IV.**

- (1) Unit Testing.
- (2) System Testing.

(3) Integration Testing.

e. **Phase V.**

(1) Project Final Report.
(2) User Manual.
(3) Interface and functionality explanation.

Chapter 2: Literature Review

2.1. Literature Review

The modern armies engaged in todays battle operations employ cutting-edge software to achieve supremacy. The concept of Network Centric Warfare (NCW) is not new, but it has been pursued since its inception. The concept itself has been proved to be valuable for the development of such kinds of applications. One of the most famously used of these software are “Blue Force Tracking”, employed by United States Army and is a successful successor to implement the NCW.

2.2. Blue Force Tracking

Blue Force Tracking is a United States military term for a GPS-enabled system that provides military commanders and forces with location information about friendly (and despite its name, also hostile) military forces. In NATO military symbology, blue typically denotes friendly forces. The system provides a common picture of the location of friendly forces and therefore is referred to as the "Blue Force" tracker.

2.2.1. System

Blue Force Tracking systems consist of a computer, used to display location information, a satellite terminal and satellite antenna, used to transmit location and other military data, a Global Positioning System receiver (to determine its own position), command-and-control software (to send and receive orders, and many other battlefield support functions), and mapping software, usually in the form of a geographic information system (GIS), that plots the BFT device on a map. The system displays the location of the host vehicle on the computer's terrain-map display, along with the locations of other platforms (friendly in blue, and enemy in red) in their respective locations. BFT can also be used to send and receive text and imagery messages, and Blue Force Tracking has a mechanism for reporting the locations of enemy forces and other battlefield conditions (for example, the location of mine fields, battlefield obstacles, bridges that are damaged, etc.).

2.2.2. Users

Users of BFT systems include the United States Army, the United States Marines Corps, the United States Air Force, the United States Navy ground-based expeditionary forces (e.g., NSWC and NECC units), and the United Kingdom.

In 2008, work began on plans to reach the level of nearly 160,000 tracking systems in the US Army within a few years; the system prime contractor is the Northrop Grumman corporation of Los Angeles, California. The M1A1 Abrams's AIM refurbishment/upgrade program includes FBCB2 and Blue Force Tracking.[1]

In November 2010 the United States Army and the United States Marines Corps reached an agreement to standardize on a shared system, to be called "Joint Battle Command Platform", which will be derived from the Army's FBCB2 system that was used by the United States Army, the United States Marines Corps, and the British Army during heavy combat operations in Iraq in 2003.

2.3. Technologies

2.3.1 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms or WPF applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

2.1.2.1.1 Features

Microsoft Visual Studio hosts these important features:

(a) Code Editor.
(b) Debugger.
(c) Designer.

2.1.2.1.2 Supported Products

Following products are supported, with extensions:

(a) Microsoft Visual C++.
(b) Microsoft Visual C#.
(c) Microsoft Visual Basic.

(d) Microsoft Visual Web Developer.
(e) Team Foundation Server.

2.3.2 Windows CE

Microsoft Windows CE is an operating system developed by Microsoft for embedded systems. Windows CE is a distinct operating system and kernel, rather than a trimmed-down version of desktop Windows.Its distinct kernel is known as “Monolithic” and “Hybrid”.

2.3.3 Google Maps API

Google launched the Google Maps API in June 2005 to allow developers to integrate Google Maps into their websites. It is a free service, and currently does not contain ads, but Google states in their terms of use that they reserve the right to display ads in the future.

2.3.4 PHP & MySQL

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers.PHP code is interpreted by a web server with a PHP processor module which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface capability and can be used in standalone graphical applications.

MySQL is the world's most widely used open source Relational Database Management System (RDBMS) that runs as a server providing multi-user access to a number of databases.The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements.

2.4. Technical Terminologies

2.4.1 Situation Awareness

Situation awareness is a broad term that covers passing GPS based updates and information gathered from sensors of registered Armor Fighting Vehicles (AFV) to respective management centers (SQN level) and Central Command and Control Center (Regiment HQ level), real-time status updates and generation of triggers for matters of concern and operational tracking in terms of troops movement and operational employment.

2.4.2 AFV Troops' Management

This concept covers records of troops' location (e.g. FUP, BOF) and replaying vehicles routes (in case of approach march etc.), reports and returns with data fusion and consolidation in terms of packet delivery services and most importantly pre-formatted and format free messages for passing information.

2.4.3 Command and Control Console

The term covers adding / deleting of registration of AFV, troops in the system, creation of multiple operational domains and associate different troops to appropriate operational domain and situational alerts.

2.4.4 Geographical Information System

This concept is related solely to maps and their usage including multiple layers of electronic maps, mapmanipulation features like zoom in/out, pan, rotate etc. and navigational assistance with distance / speed / directional calculations.

2.4.5 Packet Manager

The core idea behind BIMS, that encompasses managing data packets and checks for redundancy and delivery, load sets of packets associated to a specific operational domain, creates, manages and manipulates reports based upon operational information and last but not the least controls the GUI.

Chapter 3: System Requirements

3.1. Functional Requirements

3.1.1 Packet Manager

3.1.1.1 Description and Priority

Sub-system packet manager which acts as a core module of the system will have characteristics like manage packets and checks for redundancy and delivery, load sets of packets associated to a specific operational domain, creates, manages and manipulates reports based upon operational information and controls the system interface (GUI). Since Packet Manager Serves as the system core it is prioritized as high priority module with a rating of 9.

3.1.1.2 Stimulus/Response Sequences

Step 1. Receives and records Tank Commander operationalinformation and makes it available to the troop leader.

Step 2. Receives and records troop leader's report basing upon theinformation passed and transfers it to the interface.

Step 3. Squadron Commander receives information and assesses it and suggests action basing upon the operation input provided through interface.

Step 4. Commander at command and control level approves or disapproves the suggested action of the squadron command keeping in view the entire operational scenario.

Step 5. Basing upon the user authentication operationalinformation is made available to the intended user.

Step 6. Basing upon the user group GUI controls are made available.

Step 7. Pre-formatted messages passed at any level are interfaced to the system depending upon the message types.

Alternative 5 (Condition: User demands more information than the privilege level). Step 4 is initiated and the approval is granted basing upon situation.

Alternative 7 (Condition: Corresponding message not found in the Database.). Step 7. Check for error and go back to Step 7.

3.1.1.3 Functional Requirements

REQ001: System should allow only authenticated user to update/access and manipulate data.

REQ002: System should allow users to access data depending upon their authorization.

REQ003: System should load GUI basing upon user privileges.

REQ004: System should confirm message delivery.

REQ005: System should load associated features, sensor input and troop's situation/location at all times of operation.

3.1.2 Situation Awareness

3.1.2.1 Description and Priority

Situation Awareness (SA) covers following major responsibilities associated with this sub-system: (1) Passing GPS based updates and information gathered from sensors of registered armor fighting vehicles (AFV) to respective management centers (SQN level) and central command and control center (Regiment HQ level), (2)Real-Time status updates and generation of triggers for matters of concern, and (3)Operational tracking in terms of troops' movement and operational employment. This functionality has a priority rated at 8.

3.1.2.2 Stimulus/Response Sequences

Step 1. Sensor input and other operational information is passed,registered and processed to respective HQ.

Step 2. Trigger generation depending upon the information provided.

Step 3. Troop's location update basing upon the sensor input.

3.1.2.3 Functional Requirements

REQ006: System should allow only authenticated user to update/access and manipulate data.

REQ007: System should allow users to access data depending upon their authorization.

REQ008: System should load GUI basing upon user privileges.

REQ009: System should confirm message delivery.

REQ010: System should load associated features, sensor input and troopssituation/location at all times of operation.

3.1.3 Adding a Sensor

3.1.3.1 Description and Priority

The user shall choose to add a sensor through menu in the UI of the system. When the user clicks the button, the user shall be prompted to choose from a list of available sensors. The user shall be able to cancel the operation. (Priority: 9)

3.1.3.2 Stimulus/Response Sequences

3.1.3.2.1 Normal Flow of Events

Step 1. User clicks add sensor button on the menu.

Step 2. User is prompted to choose the list of available sensors.

Step 3. User clicks on the desired sensor to activate it.

Step 4. The sensor is activated and ready to be used for exchange of information.

3.1.3.2.2 Alternate Flow of Events

Step 1. User clicks cancel before selecting the sensor.

Step 2. The operation is canceled.

3.1.3.3 Functional Requirements

REQ011: The system shall check the availability and readiness of the sensor, which the user has selected.

REQ012: Once the sensor is selected, the system shall immediately establish connection with the sensor.

REQ013: The system shall handle cancellation properly.

3.1.4 Toggling Layers' Visibility

3.1.4.1 Description and Priority

The user shall be able to toggle visibility of a map layer (ON/OFF) by checking the checkbox for each respective map layer. (Priority: 5)

3.1.4.2 Stimulus/Response Sequences

3.1.4.2.1 Normal Flow of Events

Step 1. User clicks the map layers button to see a pop-up menu of available layers.

Step 2. User clicks show/hide checkbox of the desired map layer.

Step 3. The system makes the layer visible/invisible on the map.

3.1.4.2.2 Alternate Flow of Events

Step 1. User clicks cancel before selecting the layer.

Step 2. The operation is canceled.

3.1.4.3 Functional Requirements

REQ014: The system shall check the availability of the layer in the database, which the user has chosen to be visible.

REQ015: Once the layer is selected, the layer shall be visible over the digitized map in the system UI.

REQ016: The system shall handle cancellation properly.

3.1.5 Generating / Updating Location of Troops on the Map

3.1.5.1 Description and Priority

The module is responsible to generate location of the target by using the states of the sensors. Module receives the states of the real sensors. From only this information module generates the current location of the target. (Priority: 9)

3.1.5.2 Stimulus/Response Sequences

3.1.5.2.1 Normal Flow of Events

Step 1. Triggered event sends the state of the sensor(s) to the module.

Step 2. Module generates the current location of the target.

Step 3. The system updates the current location of the target on the map.

3.1.5.2.2 Alternate Flow of Events

Step 1. User clicks manual update mode for updating the map.

Step 2. The module shall work as and when required by the user.

Step 3. Map is updated accordingly.

3.1.5.3 Functional Requirements

REQ017: The system shall check the communication with the sensor.

REQ018: Once the data is received by the module, calculations shall be performed.

REQ019: The system shall apply the results by updating the location of troops on the map.

REQ020: The system shall handle real-time/manual updating of the map.

3.1.6 Report Generation

3.1.6.1 Description and Priority

This feature enables the user to view all relevant information of the troops and mechanized columns. The use shall click at the Generate Report button from the menu and shall be prompted with a form asking for specific information about the report to be generated. The user shall provide the start and end time of the time interval he/she wants to generate the report for, and the format of the report document. A document containing all of the relevant information such as map, sensor locations, state of troops, administrative reports etc. retrieved by the database will be generated. (Priority: 6)

3.1.6.2 Stimulus/Response Sequences

3.1.6.2.1 **Normal Flow of Events**

Step 1. User clicks generate report button on the menu.

Step 2. User is prompted to choose date and time of the report.

Step 3. User is prompted to choose the format and type of the report.

Step 4. The relevant information is retrieved by the database.

Step 5. The information is shown in the desired format of the report.

Step 6. User is prompted to save/print the report.

3.1.6.2.2 **Alternate Flow of Events**

Step 1. User discards the report.

Step 2. Report is deleted from the database and the system.

3.1.6.3 Functional Requirements

REQ021: The system shall check the validity of the date/time interval the user has selected to view a report for.

REQ022: The system shall have all the report formats available for the purpose.

REQ023: The system shall be able to store/retrieve/delete the generated reports from the database.

3.2. Non-Functional Requirements

3.2.1 Performance Requirements

(a) The Troop level sub-system will shall not exceed 1 request simultaneously.
(b) The Squadron level Sub-system shall not exceed 3 requests simultaneously.
(c) The Battalion level Sub-system shall not exceed 10 requests simultaneously.
(d) The Brigade level Sub-system shall not exceed 50 requests simultaneously.
(e) High priority messages shall be entertained if the threshold of the system is reached.

3.2.2 Safety Requirements

(a) System must not violate any law or the rights according to the military law.
(b) System must not include functionality that proxies, requests or collects un-authorized data.
(c). System must not circumvent (or claim to circumvent) intended limitations on core message passing functionality.
(d) System must verify user through two level verification i.e user ID and session ID.
(e) Applications allow user to explore extended functions like 3-D imaging, finding relief and lay of ground etc.

3.2.3 Security Requirements

(a) **Responsibility for Content**. System is responsible for all content of and within it, including advertisements, user-generated content, and any content hosted, streamed or otherwise delivered to users by third parties. Application must make it clear that this content is not provided by product.

(b) **Prohibited Content** – Application should not promote, or provide content referencing, facilitating, containing or using, the following:

(1) Barred data.
(2) Access to sensitive location.
(3) Un-ethical use of the product.
(4) Illegal activity and/or illegal contests.
(5) Content that is against any Military policy.

3.2.4 Software Quality Attributes

(a) System should not be flexible as the preformatted messages are to be passed there will be no change in any standard at all.
(b) System will be available at all times i.e.
  (1) Backup battery will be installed (the system will always be connected to the AC power supply).
  (2) As soon as the threshold is reached, System shall prioritize the messages based on the priority number.
(c) System should be reliable.
(d) System should secure and provide confidentiality of all the data.
(e) System should be portable i.e. it should be running on tablets and PCs.
(f) System should be interoperable i.e. Infantry teams can share data with armor teams in case of joint operation.
(g) Should under no circumstances allow any misuse of data.

3.2.5 Business Rules

This project is intended for the Pakistan Army therefore no business advertisement and exhibition shall be done in any circumstances.

3.3 Design and Implementation Constraints

The software will be communicating on peer-to-peer level so memory and processing power is an issue, but not much affective. Storage capacity (RHDDC and Map Databases) shall be maintained carefully. It will be accessed through Vehicular Ad-hoc Network (VANET) over SDR. It will use internet protocols such as HTTP and UDP for communication. Communication will be made secure using internet security conventions. It will be developed using C# and .NET.

Chapter 4: Software Design Specification

4.1. Introduction

The aim of our project is to develop a web based application which can be used to create, manage and communicate information related to the mechanized columns and fighting elements in the battlefield. The application allows the user to stay abreast with the situation and needs of the troops in the battlefield. And of course, users can also communicate with the troops on ground via preformatted messages and monitoring their movement and displacement on a map provided through interactive Google Maps. This project focuses on developing an application that is developed using Java which is to be used in parallel with the web based System Interface. Once the user starts the application he has an option for log-in or register. Once the user is logged-in he can view troops/mechanized columns in his area of responsibility whose locations are retrieved from GPS sensors and displayed as place markers on Google Maps.The user can also pick any element of ongoing operation from the map or from the list of available elements/mechanized columns/troops provided from the database.

4.2. Product Scope

A client server network will be set up using the network settings of SDR. Two applications will be deployed; a server-side BIMS application and a client-side BIMS application, both with slight differences. The server-side application will be available to higher commanders like Squadron Commander(s). It will feature all the functionality discussed in previous pages. The client-side application will be available to troop commanders to communicate and share information with higher commanders using server-side application.

4.3. Design Decisions

4.3.1 Design Architecture

MVC Architectural Pattern is used for this project. MVC is used because of the project nature which requires view i.e. the app interface, needs to be separated from the back end app logic so that the back end complex logic is transparent from the user of the app and he/she finds it easy to use the app.

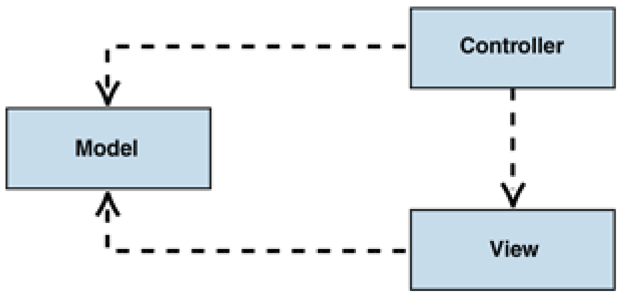


Figure-4.1 MVC Architecture

4.3.1.1 Model

The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller) i.e. the data sources and their behavior.

4.3.1.2 View

The view manages the display of information i.e. the system interface.

4.3.1.3 Controller

The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate i.e. the application logic.

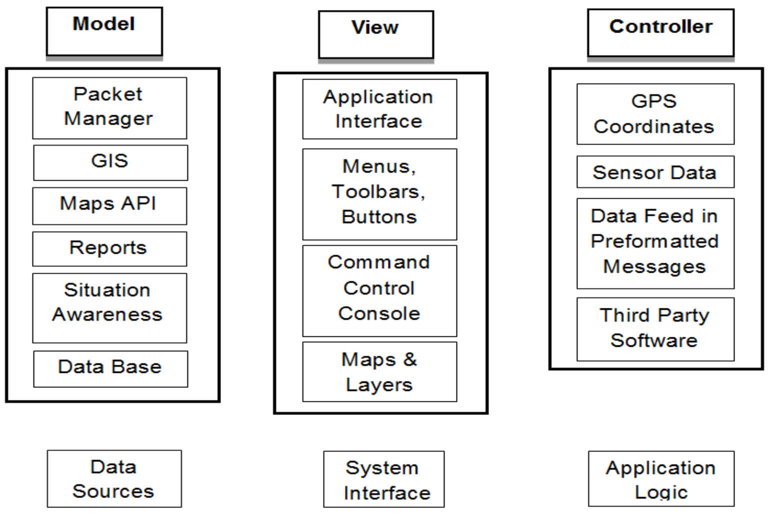


Figure -4.2 MVC Architecture for BIMS

4.3.2 Design Pattern

The design pattern used here is **Façade Pattern**. It provides a unified interface to a set of interfaces in a subsystem and defines a higher-level interface that makes the subsystem easier to use. Since, third party and proprietary software are involved in this project, therefore Façade pattern fulfills the desired functionality specific to our system interface. Our main system interface will be published amalgamating the proprietary software interfaces as well as other sub-system interfaces like GIS, Command and Control Console etc.

4.3.3 Use Case Diagrams

4.3.3.1 System Use Case

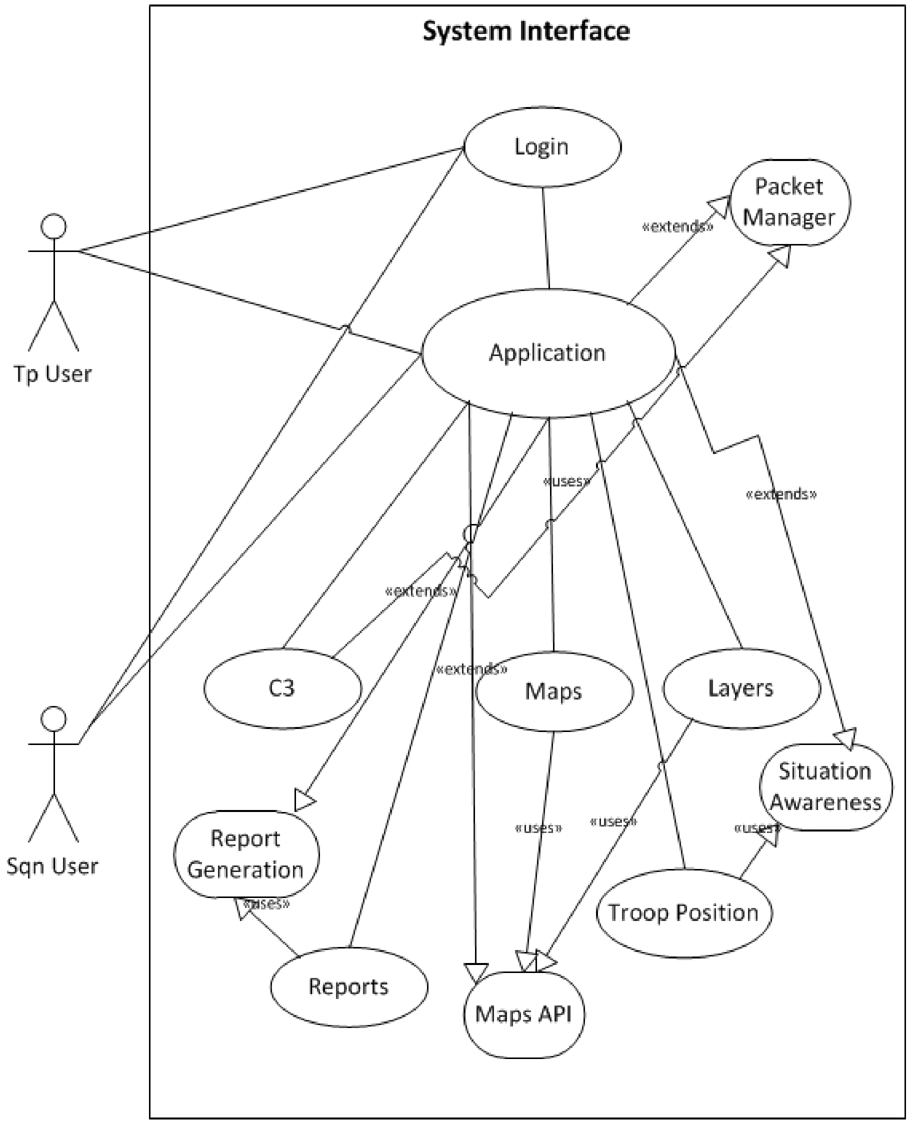


Figure - 4.3 System Use Case

4.3.3.2 Login Use Case

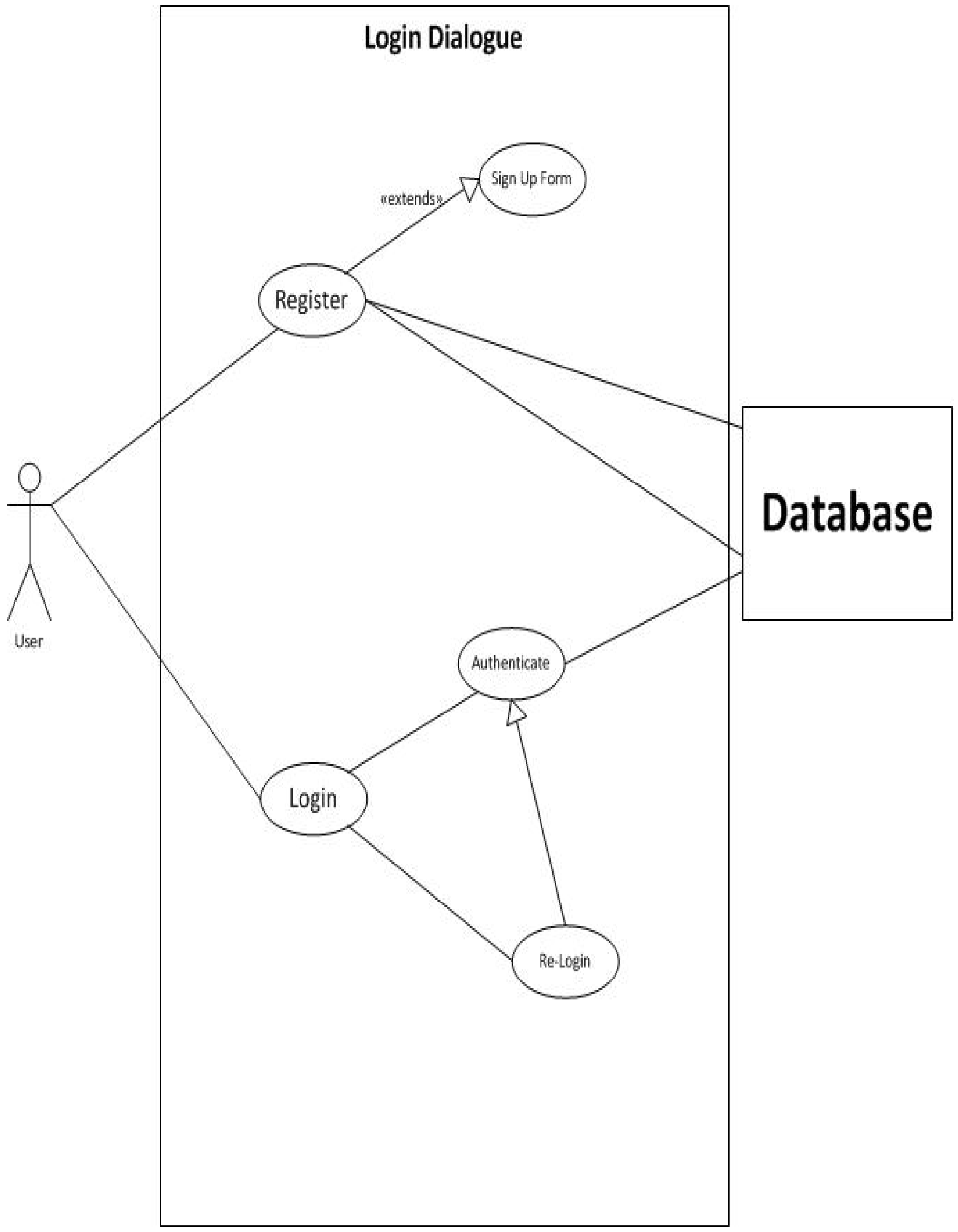


Figure – 4.4 Login Use Case

4.3.3.3 GPS Sensors and Coordinates Use Case

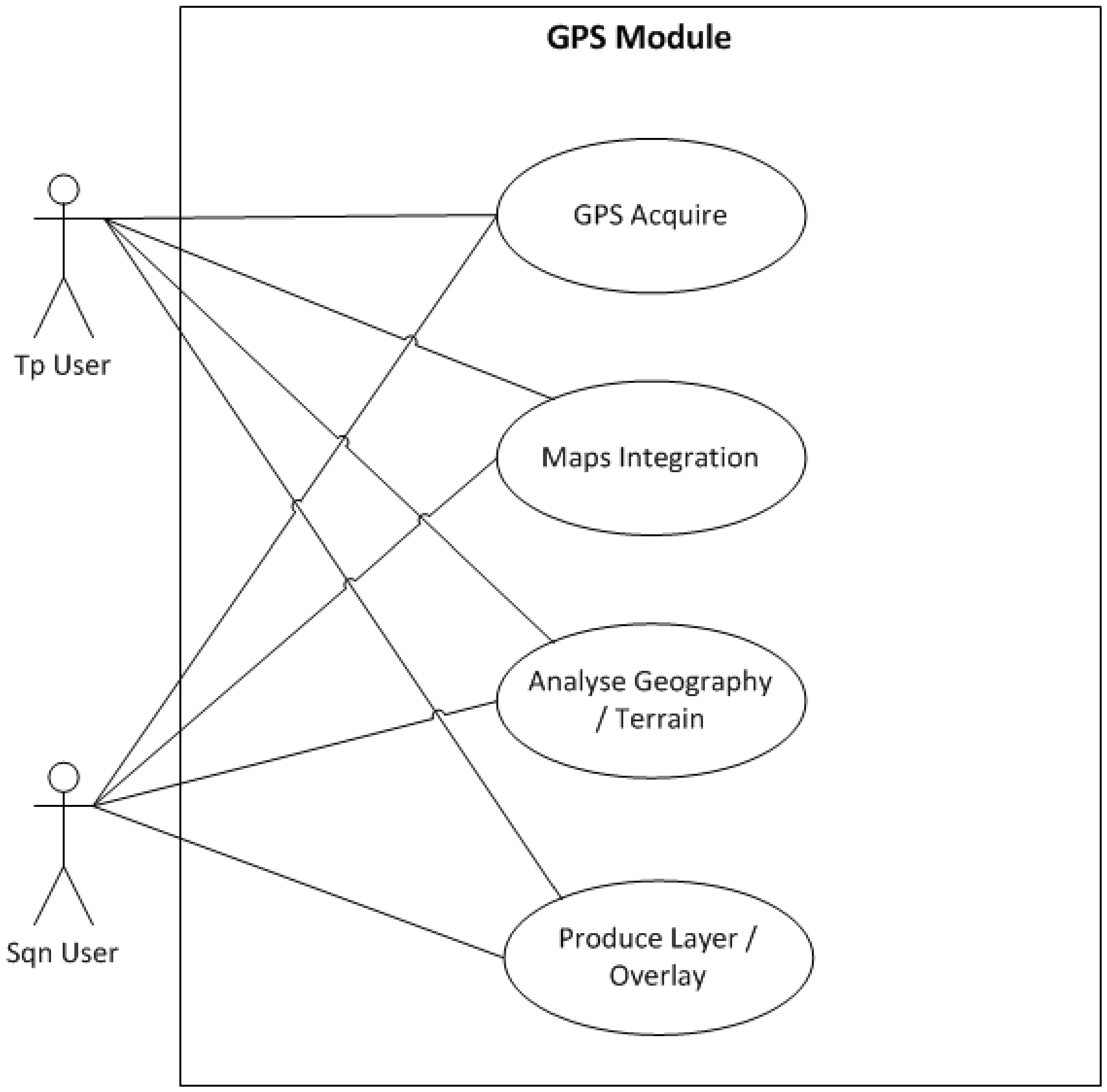


Figure – 4.5 GPS Module Use Case

4.3.3.4 Packet Manager Use Case

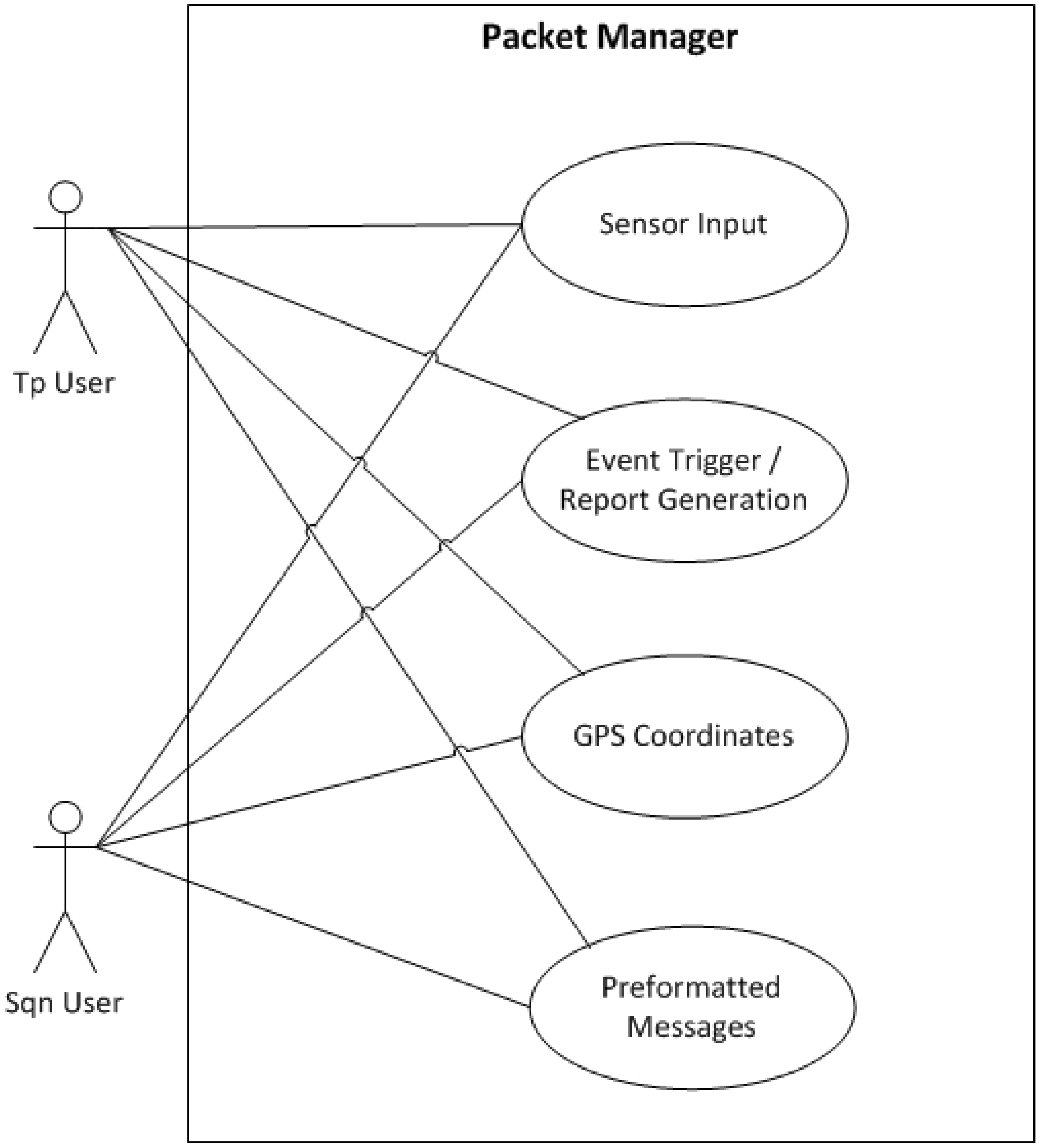


Figure – 4.6 Packet Manager Use Case

4.3.4 Class Diagram

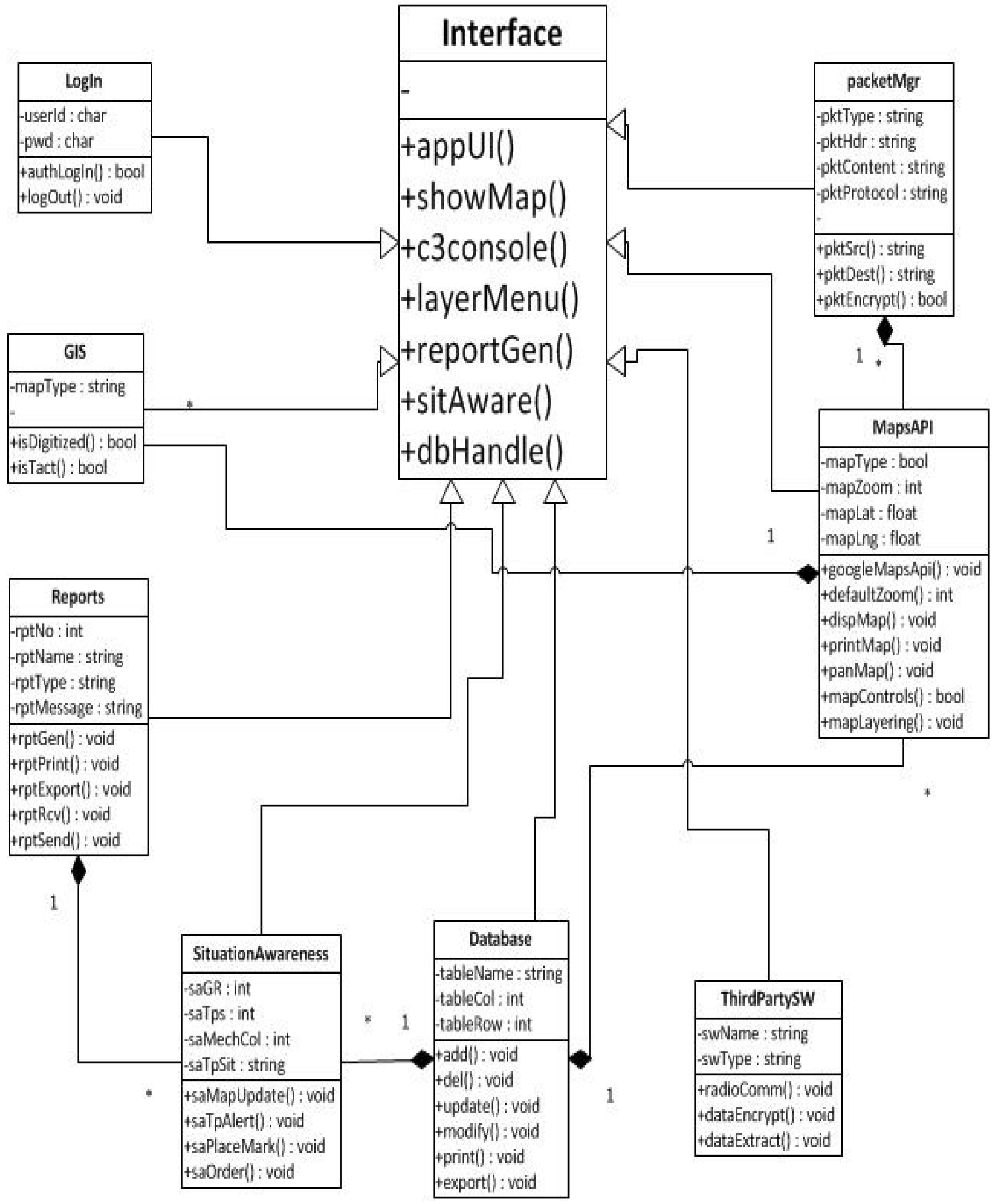


Figure – 4.7 System Interface Class

4.3.5 Class Diagram Details

4.3.5.1 Interface Class

4.3.5.1.1 Description

The main class of the system interface, which will handle all the communication and utilize the functionality of its sub-systems.

4.3.5.1.2 Variables

May include variables during development.

4.3.5.1.3 Functions

appUI():	To integrate/display all the information.
showMap():	To update and display maps.
c3console():	For utilizing command messages.
layerMenu():	To use map layers of various types.
reportGen():	To generate reports on need basis.
sitAware():	To update/display troop deployments etc.
dbHandle():	To handle requests to/from databse.

4.3.5.2 Login Class

4.3.5.2.1 Description

The class deals with the login of a user, and verifying the credentials of the user.

4.3.5.2.2 Variables

userId:	Username of the BIMS user.
pwd:	Password provided by the user.

4.3.5.2.3 Functions

authLogin() &logOut() for authentication and logout of users.

4.3.5.3 GIS Class

4.3.5.3.1 Description

Geographic Information System class. Mainly deals with the type of map involving ground features, contours and other details that define the terrain of the area of operation.

4.3.5.3.2 Variables

mapType:	Stores the type of the map to be used as base.

* May include more variables during development.

4.3.5.3.3 Functions

isDigitized() &isTact() check the type of the map used.

4.3.5.4 Reports Class

4.3.5.4.1 Description

One of the major classes for generating reports of all sort. May host features to produce statistical data, administrative and operational reports.

4.3.5.4.2 Variables

rptNo:	Number assigned to a report.
rptName:	Name assigned to a report.
rptType:	Type of report to be generated.
rptMessage:	Message contained in the report.

4.3.5.4.3 Functions

rptGen():	To generate a report.
rptPrint():	To print hard copy of the report.
rptExport():	To export a report in other formats.

rptRcv() &rptSend():For email/fax etc.

4.3.5.5 packetMgr Class

4.3.5.5.1 Description

The core class of the BIMS that deals with the transfer of information between interface and the hardware.

4.3.5.5.2 Variables

pktType:	Standard network packet type identifier.
pktHdr:	Header information of a packet.
pktContent:	Contents of message to be transferred.
pktProtocol:	Protocol to be used for transferring.

4.3.5.5.3 Functions

pktSrc():	Originator's address of the packet.
pktDest():	Address of the packet receiver.
pktEncrypt():	Whether to encrypt a packet or not.

4.3.5.6 MapsAPI Class

4.3.5.6.1 Description

Main class for manipulating and handling the map requests.

4.3.5.6.2 Variables

mapType:	Types of maps provided by Google Maps.
mapZoom:	To adjust zoom level of the map.
mapLat:	Latitude coordinate of the default location.
mapLng:	Longitude coordinate of the default location.

4.3.5.6.3 Functions

googleMapsAPI():	Standard API provided by Google Maps.
defaultZoom():	Default zoom level to display a map.
dispMap():	Main function to display the map::showMap().
printMap():	To get a printout of a map.
panMap():	To change map handling mode.
mapControls():	Zoom/pan/type etc. modes of the map.
mapLayering():	Layers available for a specific map.

4.3.5.7 SituationAwareness Class

4.3.5.7.1 Description

An important class to make use of maps and display location of troops in theatre of operations.

4.3.5.7.2 Variables

saGR:	Grid references of the troop(s) location.
saTps:	Details of the troops concerned.
saMechCol:	State of the mechanized columns involved.
saTpSit:	Situational state of troop(s).

4.3.5.7.3 Functions

saMapUpdate():	Update location(s) of troops on map.
saTpAlert():	Emergency messages of troops.
saPlaceMark():	Markers indicating troops' position.
saOrder():	For delivering order messages to troop(s).

4.3.5.8 Database Class

4.3.5.8.1 Description

A class to handle the queries of database.

4.3.5.8.2 Variables

tableName:	Name of the desired table (in case of new).
tableCol:	Number of columns of the desired table.
tableRow:	Number of rows of the desired table.

4.3.5.8.3 Functions

add(), del(), update(), modify(), print(), export(): Standard functions of table operations on queries.

4.3.5.9 ThirdPartySW Class

4.3.5.9.1 Description

A class to handle communication with third party software associated with the radio sets being used. They're in fact proprietary software.

4.3.5.9.2 Variables

swName:	Name of the software to interact with.
swType:	Type/Nature of software being used/called.

4.3.5.9.3 Functions

radioComm():	To send/receive commands to radio set.
dataEncrypt():	Whether to encrypt comm or not.
dataDecrypt():	Whether to decrypt comm or not.
dataExtract():	Extract any useful/technical data.

4.3.6 Sequence Diagrams

4.3.6.1 Login Sequence

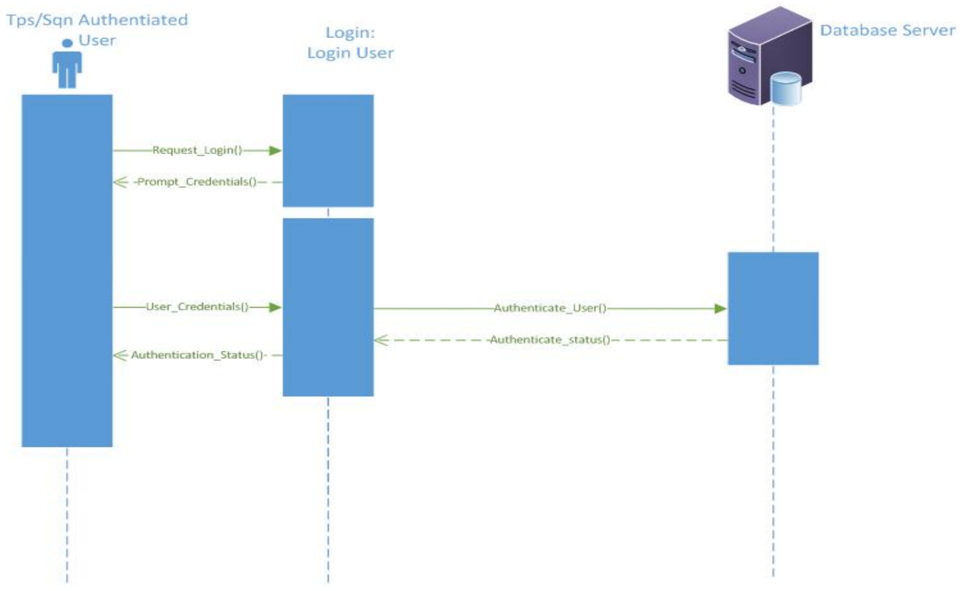


Figure – 4.8 Login Sequence

4.3.6.2 Maps View / Update Sequence

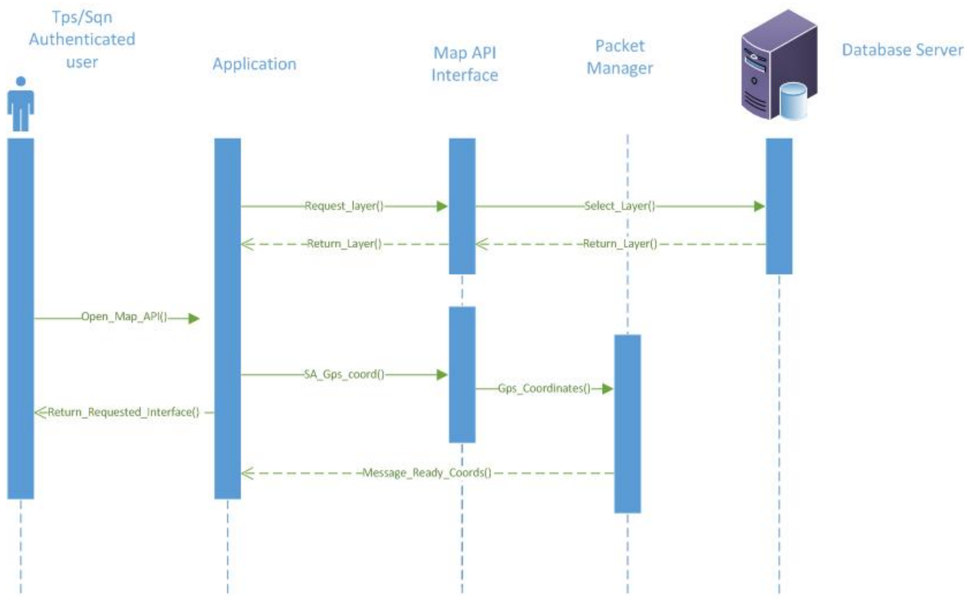


Figure – 4.9 Maps View/Update Sequence

4.3.6.3 Preformatted Messages Sequence

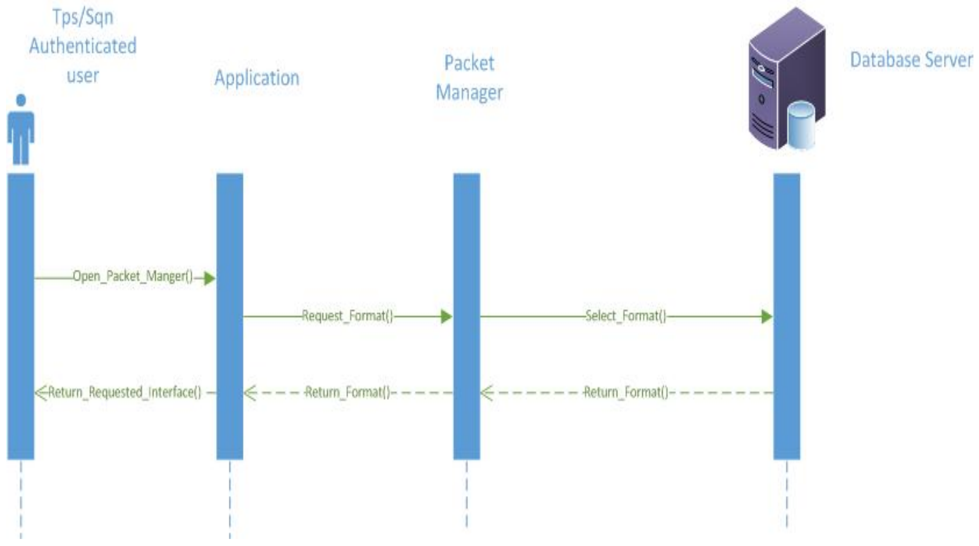


Figure – 4.10 Preformatted Messages Sequence

4.3.7 Activity Diagrams

4.3.7.1 Map Layer Activity

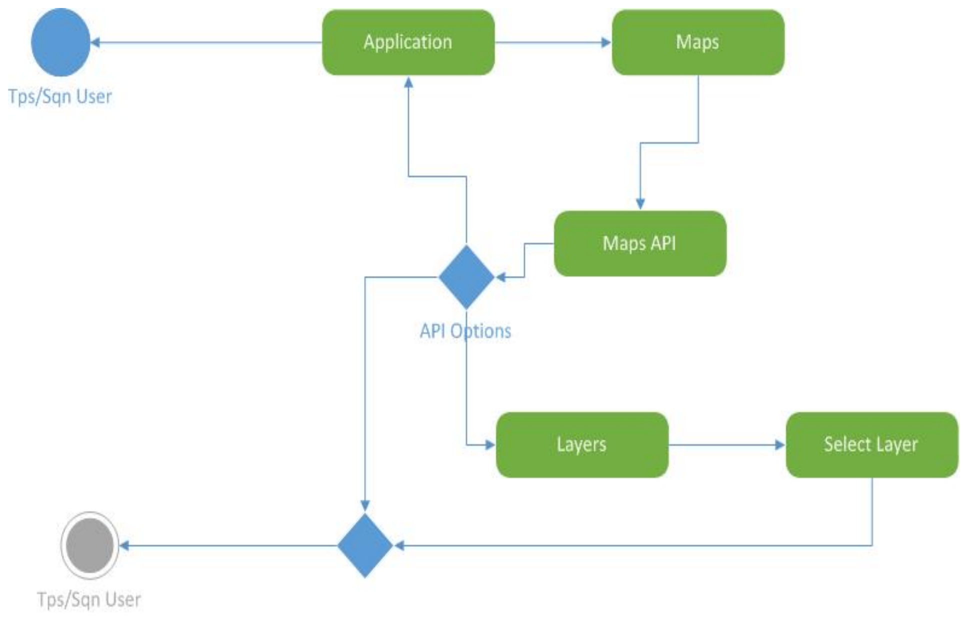


Figure – 4.11 Map Layer Activity

4.3.7.2 Preformatted Messages Activity

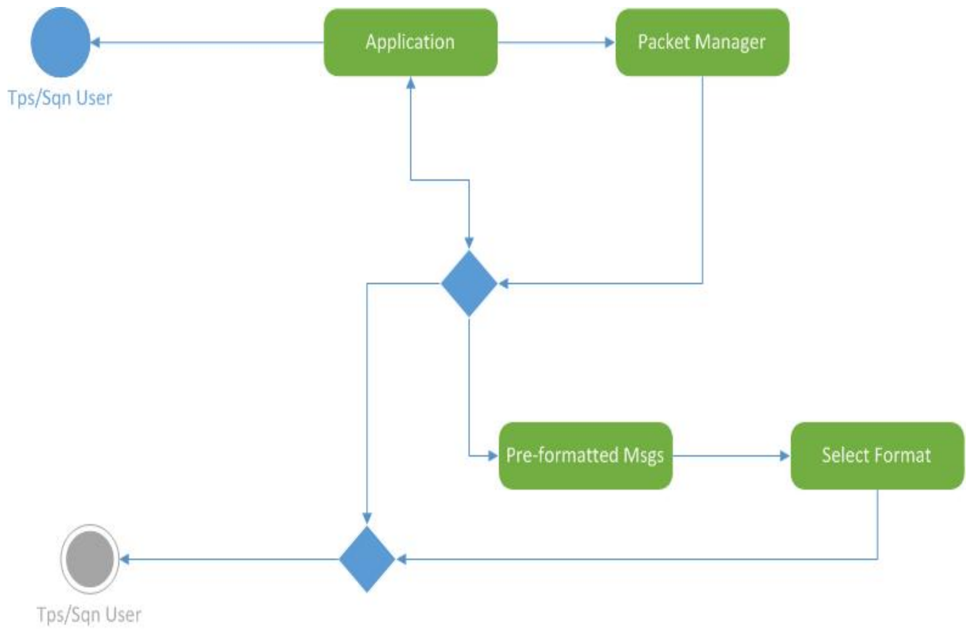


Figure – 4.12 Preformatted Messages Activity

4.3.7.3 Report Generation Activity

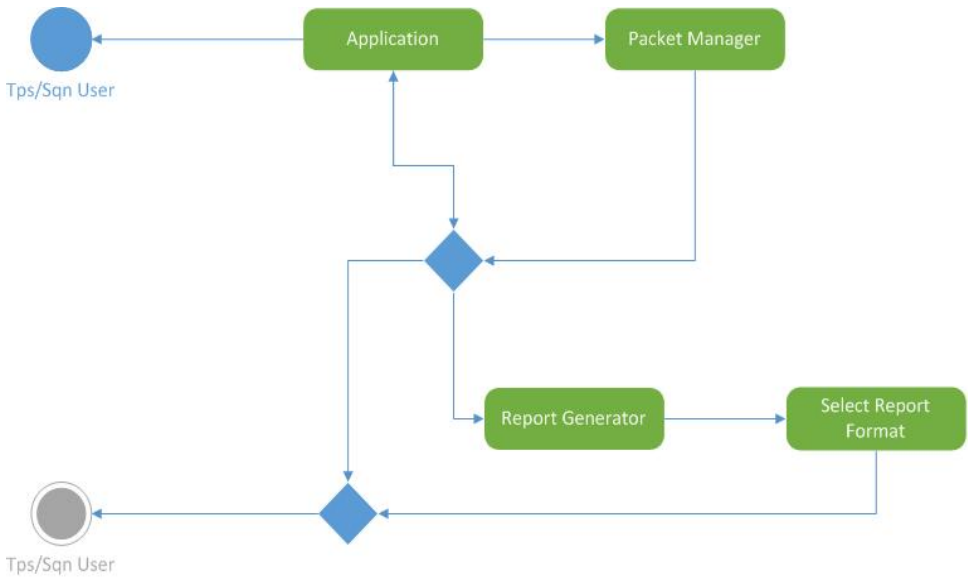


Figure – 4.13 Report Generation Activity

4.3.7.4 Situation Awareness Activity

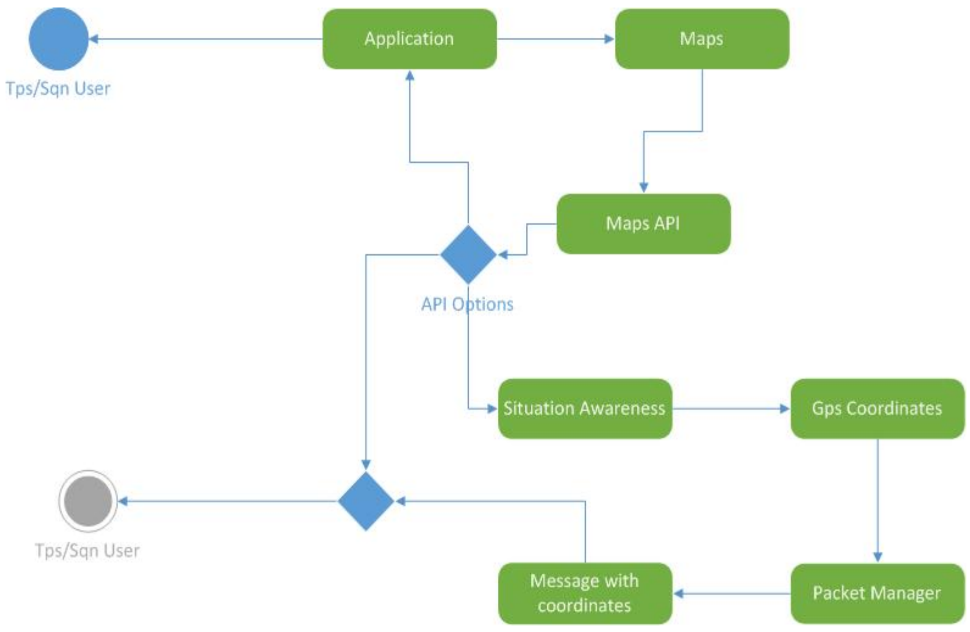
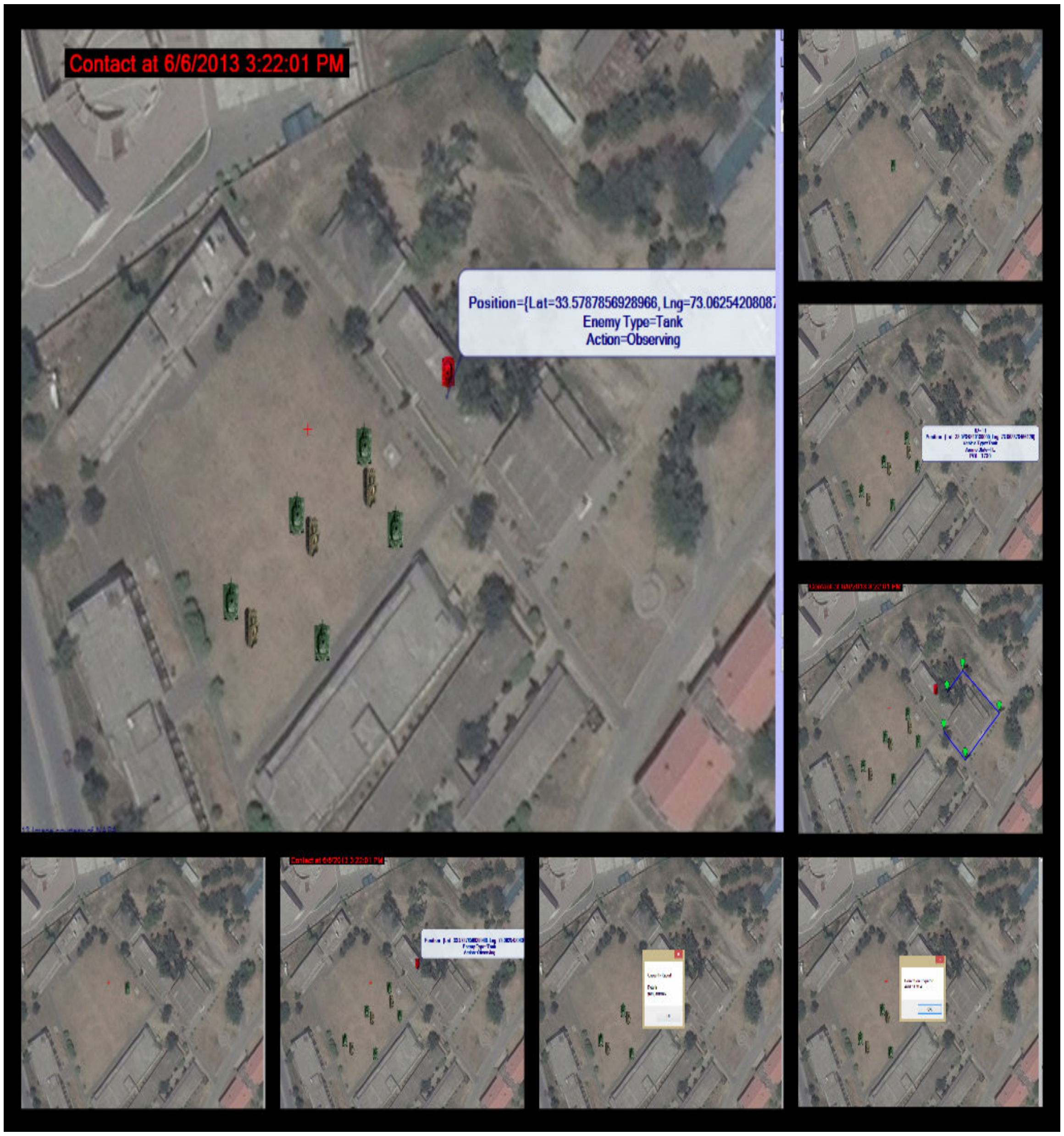


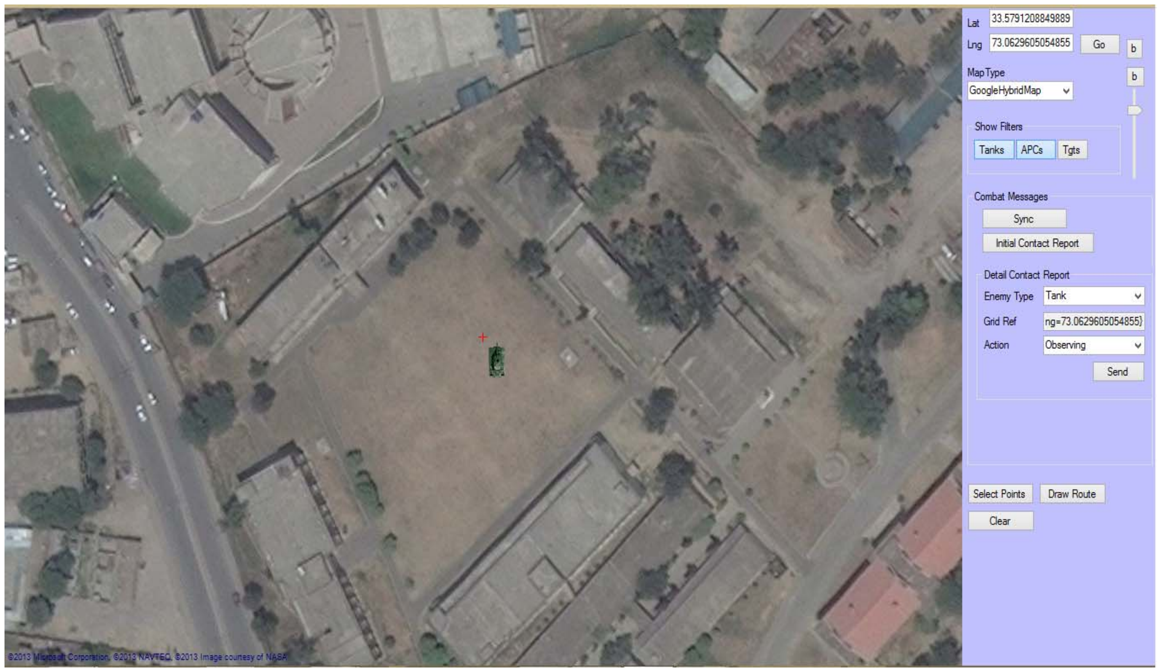
Figure – 4.14 Situation Awareness Activity

4.3.8 System Interface


Contact at 6/6/2013 3:22:01 PM
Position={Lat=33.5787856928966, Lng=73.06254208087
Enemy Type=Tank
Action=Observing

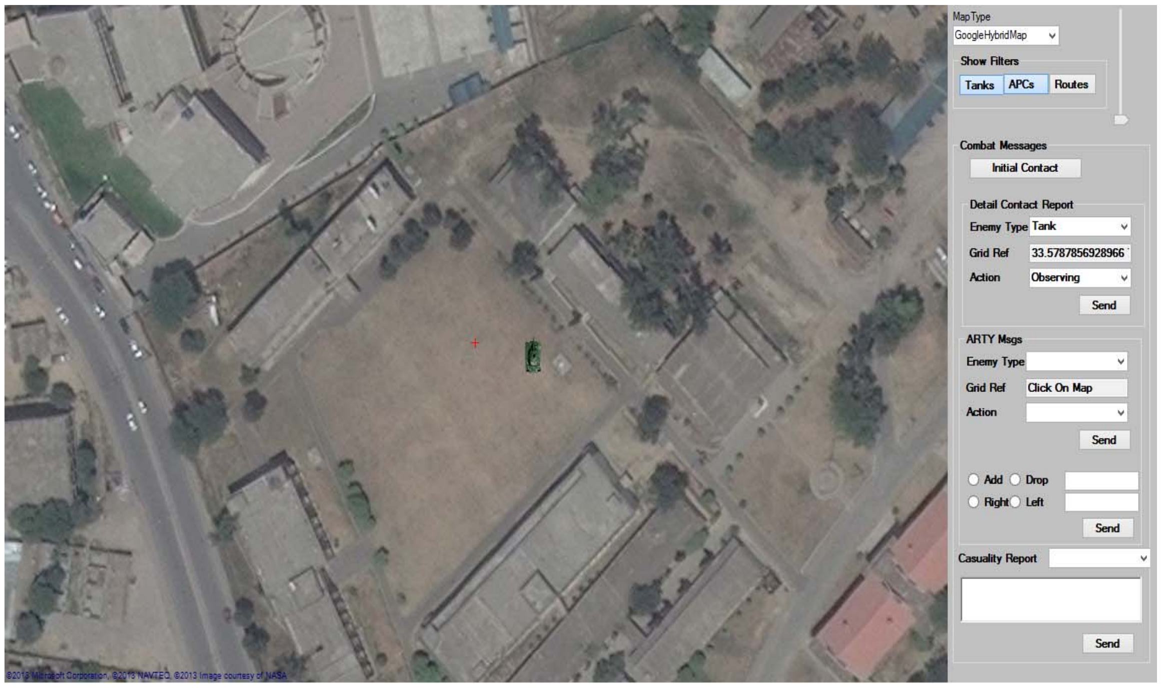
Picture – 4.1 Difference Screens of System

4.3.8.1 BIMS Server Interface


Lat 33.5791208849889
Lng 73.0629605054855
Map Type GoogleHybridMap
Enemy Type Tank
Grid Ref ng=73.0629605054855}
Action Observing

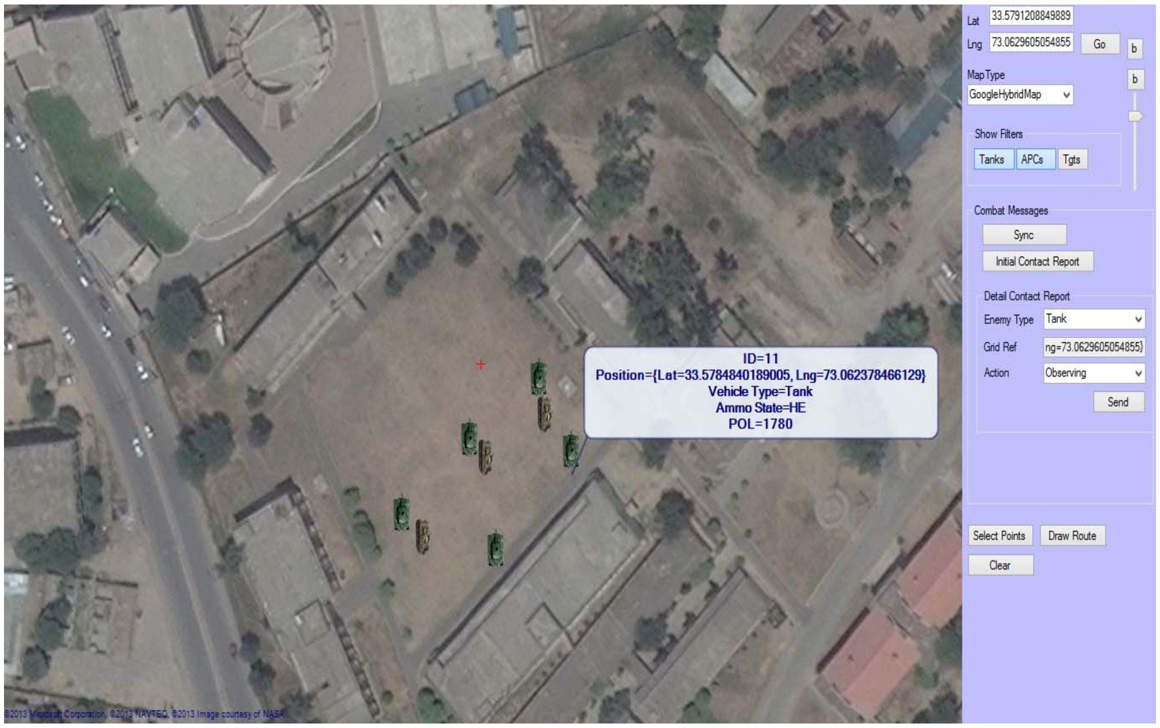
Picture – 4.2 BIMS Server Application

4.3.8.2 BIMS Client Interface


Map Type GoogleHybridMap
Enemy Type Tank
Grid Ref 33.5787856928966
Action Observing
ARTY Msgs
Enemy Type
Grid Ref Click On Map
Action

Picture – 4.3 BIMS Client Application

4.3.8.3 Server Sync Snapshot

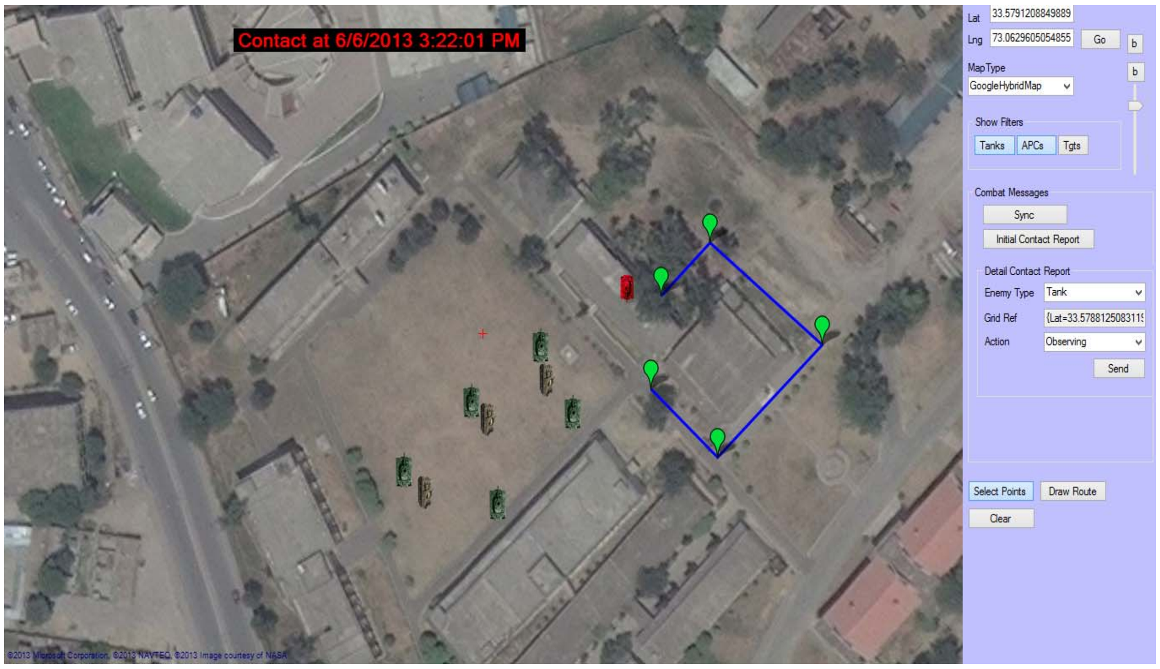


ID=11
Position={Lat=33.5784840189005, Lng=73.062378466129}
Vehicle Type=Tank
Ammo State=HE
POL=1780

Lat 33.5791208849889
Lng 73.0629605054855
MapType GoogleHybridMap
Detail Contact Report
Enemy Type Tank
Grid Ref ng=73.0629605054855}
Action Observing

Picture – 4.4 Synchronization

4.3.8.4 Server Route Snapshot

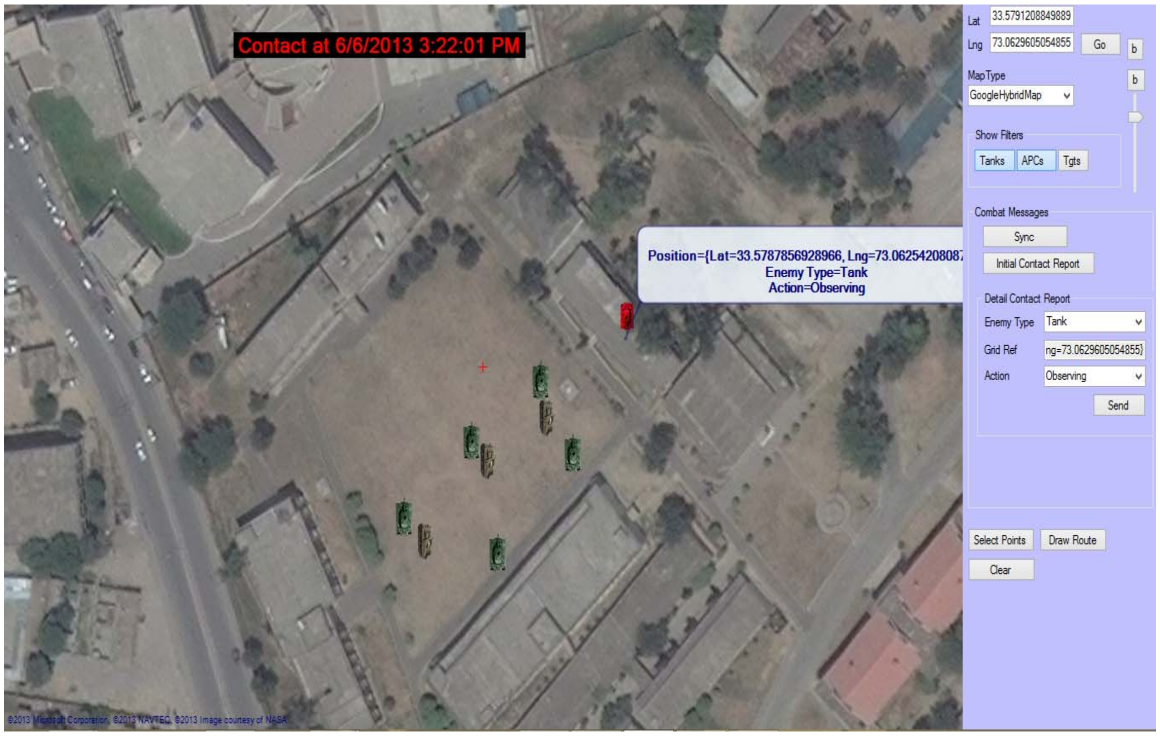


Contact at 6/6/2013 3:22:01 PM

Lat 33.5791208849889
Lng 73.0629605054855
MapType GoogleHybridMap
Detail Contact Report
Enemy Type Tank
Grid Ref {Lat=33.578812508311
Action Observing

Picture – 4.5 Route Marking

4.3.8.5 Contact Report Snapshot

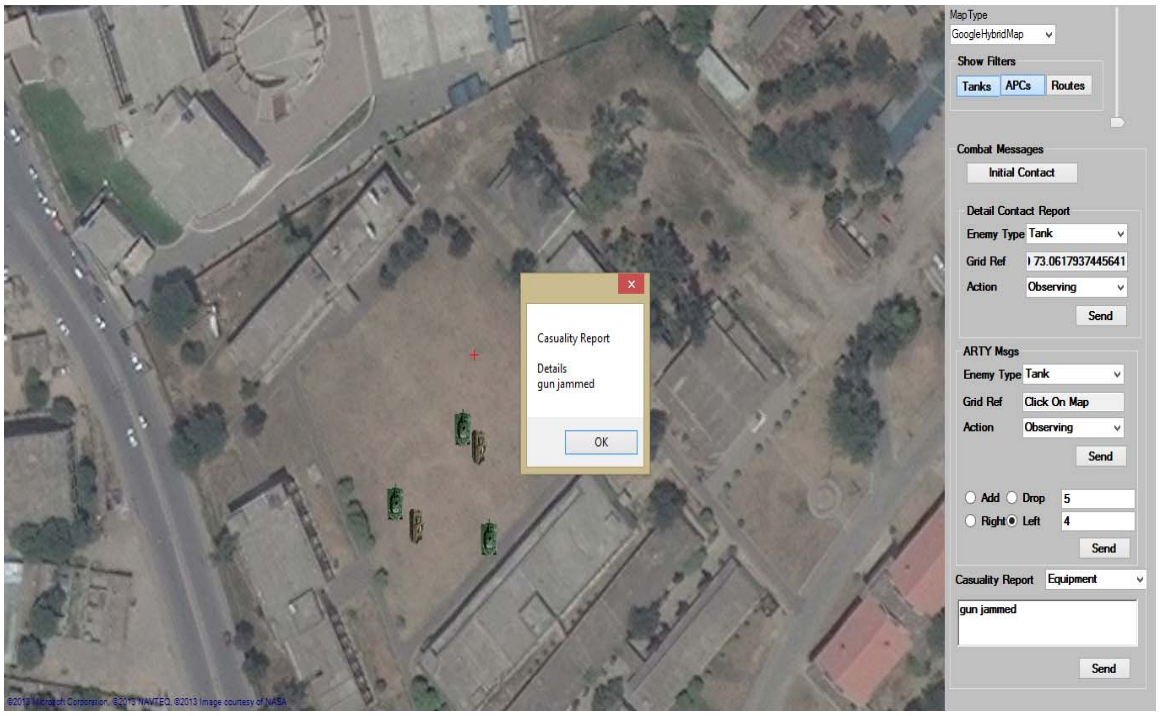

Contact at 6/6/2013 3:22:01 PM

Position={Lat=33.5787856928966, Lng=73.06254208087
Enemy Type=Tank
Action=Observing

Lat 33.5791208849889
Lng 73.0629605054855
Map Type: GoogleHybridMap
Enemy Type: Tank
Grid Ref: ng=73.0629605054855}
Action: Observing

Picture – 4.6 Contact Report

4.3.8.4 Casualty Report Snapshot


Casuality Report

Details
gun jammed

Map Type: GoogleHybridMap

Detail Contact Report
Enemy Type: Tank
Grid Ref:) 73.0617937445641
Action: Observing

ARTY Msgs
Enemy Type: Tank
Grid Ref: Click On Map
Action: Observing

( ) Add ( ) Drop 5
( ) Right (●) Left 4

Casuality Report: Equipment
gun jammed

Picture – 4.7 Casualty Report

Chapter 5: System Implementation

5.1. Introduction

5.1.1 Purpose

Battlefield Information & Management System (BIMS) is aimed at developing a system to transfer data packets over the radio communication network incorporating SDRs i.e. 9661 radio sets. Main objective of BIMS is to update the present location of moving echelons, and pass mission-critical information related to mechanized columns to the higher commanders. BIMS will combine the features of SA, ATM, C3 and GIS modules to enable the efficient use of resources within the enemy's decision cycle. And this is the main theme for the implementation of the system.

5.1.2 System Overview

Battlefield Information and Management System is intended to offer the following functions:

(a) Data Acquisition
(b) Redundancy and Delivery Checks
(c) Packet Crafting and Data Sending Module
(d) Maps Handling Application
(e) Packet Manager
(f) Graphical User Interface (GUI)

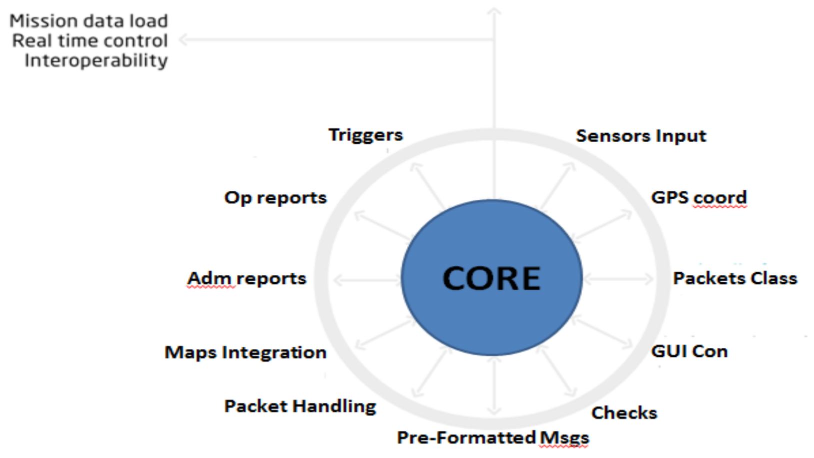


Figure – 5.1 System Overview

5.2. Implementation Considerations

5.2.1 Assumptions and Dependencies

The default settings of the system are various predefined set of variables. These variables are set in the backend code of the system, and can be changed based on the user needs. These include:

(a) Map coordinates
(b) Map zoom level
(c) Map overlays
(d) Map orientation (w.r.t North etc.)

5.2.2 Constraints

Some of the constraints that are incorporated in the system, during implementation, are as:

(a) Database of digitized maps will be used as base maps where there is no network connectivity or Google Maps not available.
(b) If, used on mobile, the user must have internet connectivity.
(c) Huge data cannot be saved on the mobile.

5.2.3 Software

(a) Microsoft Windows XP or later (for PC version)
(b) Windows CE / Android OS 2.3 or later (for Tablet version)
(c) Space requirement: 50 MB minimum (including cache for maps)
(d) Third party software provided with SDR.

5.2.4 Hardware

(a) Software Defined Radios (SDRs)
(b) Ethernet cable
(c) 2 PCs / laptops and / or 2 Tablets
(d) 2 GB RAM (Recommended)

5.2.5 Technologies

Microsoft Visual Studio 2005 & 2012, Android SDK, PHP / MySQL

5.3. Code Excerpts

5.3.1 Client-sideCode Excerpt

```
privatevoidSyncConnect()
        {
try
            {
TcpClient client = newTcpClient("127.0.0.1", 9877);
Syncns = client.GetStream();
reader.Start();
            }
catch
            {
MessageBox.Show("Unable to connect\nClosing Application");
Application.Exit();

            }
        }
privatevoid Connect()
        {
try
            {
TcpClient client = newTcpClient("127.0.0.1", 9876);
ns = client.GetStream();
msgrcv.Start();
            }
catch (Exception)
            {

            }
        }
privatevoid Reader()
        {
try
            {
while (true)
                {
msgs.Add(SyncReceiver(Syncns));
Console.WriteLine(msgs[0].Grid_Ref.ToString());

                }
            }
catch { }


        }
List<PointLatLng> poi;
privatevoidMsg()
        {
StreamReadersr = newStreamReader(ns);
char[] buffer = newchar[256];
try
            {
while (true)
                {
buffer = newchar[256];
sr.Read(buffer, 0, buffer.Length);
stringrcv = newstring(buffer);
rcv = rcv.Substring(0,rcv.IndexOf('\0'));
if (rcv == "Hello")
                    {

SyncWriter(Syncns, sm);

                    }
elseif (rcv == "SyncDone")
                    {



sync_done = true;
                    }
if (rcv == "yes")
                    {
BinaryFormatterbn = newBinaryFormatter();
poi = bn.Deserialize(Syncns) asList<PointLatLng>;
foreach (PointLatLng p in poi)
                        {
mr.Points.Add(p);
                        }

                    }
if (rcv == "no")
MessageBox.Show("No routes Drawn");
Console.WriteLine();

                }
            }
catch { }


        }
```

5.3.2 Server-sideCode Excerpt

```
privatevoid Listener()
        {
nsl = newList<NetworkStream>();
try
            {
                TcpListenersrvr = new
                TcpListener(IPAddress.Parse("127.0.0.1"),9876);
srvr.Start();
while (true)
                {
NetworkStream ns = srvr.AcceptTcpClient().GetStream();
nsl.Add(ns);
Thread read = newThread(() =>Msg(nsl.Count - 1));
read.IsBackground = true;
read.Start();
MessageBox.Show("Connected");
                }
            }
```

```
catch (Exceptioner)
            {

MessageBox.Show("cdvf");
            }

        }

privatevoidSyncListener()
        {
/*  try
            {
syncnsl = new List<NetworkStream>();
                TcpListenersrvr = new
                TcpListener(IPAddress.Parse("192.168.10.1"), 9877);
srvr.Start();

try
                {
while (true)
{ syncnsl.Add(srvr.AcceptTcpClient().GetStream());
Console.WriteLine("Connected2"); }

                }
catch { }
            }
catch (Exception)
            {

MessageBox.Show("DFDS");
            }
           * */
syncnsl = newList<NetworkStream>();
TcpListenersrvr = newTcpListener(IPAddress.Parse("127.0.0.1"),9877);
srvr.Start();
boolzz=false;
while(zz==false)

            {
try
            {
while (true)
                {
syncnsl.Add(srvr.AcceptTcpClient().GetStream());
Console.WriteLine("Connected2"); zz = true;
                }

            }
catch { }
            }
```

5.3.3 GPS Coordinates Fetching Code

```
privatevoid backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
string line;
int count = 0;


varfs = newFileStream(@"D:\\GPS_INPUT.nmea", FileMode.Open,
FileAccess.Read, FileShare.ReadWrite);

System.IO.StreamReader file =
newSystem.IO.StreamReader(fs);

while (true)
            {
if ((line = file.ReadLine()) == null)
continue;

try
                {
if (line.IndexOf("$GPGGA") == 0)
                    {
String[] matches = line.Split(',');
lat = double.Parse(matches[2]);
lat /= 100.0;
lat = Math.Floor(lat) + (lat % 1) / 60.0 * 100.0;
if (matches[3].Equals("S")) lat = -lat;
lng = double.Parse(matches[4]);
lng /= 100.0;
lng = Math.Floor(lng) + (lng % 1) / 60.0 * 100.0;
if (matches[4].Equals("W")) lng = -lng;

noti += "lat: " + lat + "\t" + "long: " + lng + "\r\n";

count++;
if (count == 10)
                        {
count = 0;
backgroundWorker1.ReportProgress(100);
                        }
                    }
                }
catch (Exceptionexcep)
                {
//
                }

if (backgroundWorker1.CancellationPending)
break;
            }

file.Close();
        }
```

Chapter 6: Testing and Results Analysis

6.1. Server Application Tests

Test Case ID	1
Unit to Test	Login
Assumptions	1. User not logged in 2. User exists 3. Login page opened
Test Data	1. Username 2. Password
Steps to be Executed	1. Visit login page 2. Enter username 3. Enter password 4. Click login button
Expected Result	Reload the application page with user logged in
Actual Result	Application page is reloaded with user logged in
Pass/Fail	Pass

Test Case ID	2
Unit to Test	Logout
Assumptions	1. User logged in 2. User exists 3. Application running with user logged in status
Test Data	User session information
Steps to be Executed	1. Click on logout button
Expected Result	Reload the application page with user logged out
Actual Result	Application page is reloaded with user logged out
Pass/Fail	Pass

Test Case ID	3
Unit to Test	Map Initialization
Assumptions	1. Application is running 2. User logged in
Test Data	1. Own position from GPS 2. Availability of maps in cache 3. Place marker of own position on map
Steps to be Executed	1. Application started by user 2. User logs in 3. Application checks for own position Grid Reference (GR) from GPS 4. Based on own GR, application checks for corresponding map tiles in cache 5. Load the map tiles from cache 6. Place marker for own position on map
Expected Result	Map is loaded from cache with own position GR marker placed on it
Actual Result	Map loaded and own position GR marker visible on it
Pass/Fail	Pass

Test Case ID	4
Unit to Test	Synchronization
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application running on all nodes
Test Data	1. Synchronization of data between server and nodes 2. All nodes updated 3. Place markers of respective own positions of all nodes on their maps 4. Place markers of all nodes on map on server 5. Response time of synchronization
Steps to be Executed	1. Server application started 2. Client application(s) started 3. Sync button clicked on all client(s) 5. Load the map tiles from cache of

	respective applications 6. Place marker(s) for own position(s) on maps of respective application(s)
Expected Result	Synchronization is successful, map grid updated on all nodes and server with each having place markers of all nodes and server
Actual Result	Synchronization successful, map grid updated with own position of all the elements in the network
Pass/Fail	Pass

Test Case ID	5
Unit to Test	Failed Nodes
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application running on all Nodes 4. Synchronization is in progress 4. Synchronization NOT successful with all nodes
Test Data	1. Resending of sync packets to failed nodes 2. Synchronization of data between server and failed nodes 3. All nodes updated 4. Place markers of respective own positions of all nodes on their maps 5. Place markers of all nodes on map on server 6. Response time of synchronization
Steps to be Executed	1. Application running on server and failed nodes 2. Failed nodes acknowledge resent sync packets 3. Application checks for own position Grid Reference (GR) from GPS 4. Based on own GR, application checks for corresponding map tiles in cache 5. Load the map tiles from cache 6. Place marker for own position on map

Expected Result	Failed nodes acknowledge sync request and update themselves
Actual Result	Maps loaded failed nodes and own position GR marker of all elements visible on them
Pass/Fail	Pass

Test Case ID	6
Unit to Test	Packet Types
Assumptions	1.Transfer of packets between server and client(s) 2. Packets bearing their type according to intended message 3. Packet(s) successfully received at either side i.e. server or client
Test Data	1. Checking of types of data packets 2. Display of information on type of data packets 3. Storage / update of information in database depending on type of data packets
Steps to be Executed	1. Client sends a report 2. Server accepts the report 3. Server checks the type of report 4. Server updates the database for corresponding record table 5. Display the alert of desired report
Expected Result	Desired report is displayed on server/client, and updated in database
Actual Result	1. Ammunition report was sent from client to server, server generated trigger to display ammunition report from client, database updated with latest information 2. Same action was performed for POL report by checking type of data packet
Pass/Fail	Pass

Test Case ID	7
Unit to Test	Initial Contact Report
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application running on all Nodes 4. All nodes in synchronization with server
Test Data	1. Sending of initial contact report to server by a client 2. Reception of initial contact report at server 3. Plotting of enemy elements on map on server based on sent GRs 4. Server sending enemy GRs to all other nodes
Steps to be Executed	1. Client sees an enemy and creates an initial contact report 2. Click send button to send report to server 3. Server updates the enemy position on map 4. Server send sync messages to all other nodes to update their maps with enemy GRs 5. All other nodes update their maps
Expected Result	Initial contact report is received at server, server updates its maps and all other nodes' maps
Actual Result	Enemy sighted by client, initial contact report was made, sent to server. Server sync with all other participating clients and updated them
Pass/Fail	Pass

Test Case ID	8
Unit to Test	Detailed Contact Report
Assumptions	1. Client-Server network established 2. Server and clients in sync 3. Initial contact report is received at server 4. Initial contact report is shared and updated with all other nodes

Test Data	1. Information contained in initial contact report 2. Validation of information 3. Generation of detailed contact report
Steps to be Executed	1. Server checks the information from other nearest client to enemy to verify it 2. Creates detailed contact report 3. Sends detailed contact report to all the nodes including actions to be taken 4. All nodes receive the update 5. Update their maps
Expected Result	Detailed contact report received at every node and maps updated
Actual Result	Server sent detailed contact report to clients and actions to be taken against the enemy
Pass/Fail	Pass

Test Case ID	9
Unit to Test	Artillery Target Grid Message
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application running on artillery observer's node 4. Synchronization done
Test Data	1. Sending of artillery target message to server 2. Reception of artillery target message at server 3. Plotting of enemy elements on map on server based on sent GRs 4. Verification of GRs from nodes nearby the enemy 5. Server sending enemy GRs to all other nodes 6. Recording of target information in database
Steps to be Executed	1. Artillery observer sends the target message to server by clicking send report button 2. Server receives the artillery target grid message 3. Server checks the information from

	other nearest client to enemy to verify it 4. Creates detailed contact report 5. Sends detailed contact report to all the nodes and the artillery observer including actions to be taken 6. All nodes receive the update 7. Update their maps 8. Server updates the database with new target information
Expected Result	Detailed contact report received at every node & artillery observer's node and maps updated, database at server updated
Actual Result	Artillery observer receives detailed contact report with all other nodes also updating their maps, and server updated the target records in database
Pass/Fail	Pass

Test Case ID	10
Unit to Test	Route Generation
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application(s) also running 4. Synchronization done 5. Intial / Detailed / Artillery target contact report sent and received at server
Test Data	1. Availability of target data in database 2. What to do if it is not available?
Steps to be Executed	1. Target data received at server. 2. Server checks for record of existing target data 3. If, target exists in database, generate and include route in detailed contact report 4. If, target does not exists in database, ask the node to generate and adopt it's own route to target 5. Synchronization of routes generated in 4 & 5 with all other nodes in the form of detailed contact report

Expected Result	1. If target existed in database, send it to all nodes 2. If target route not in database, subject node to generate it's own route and send to server 3. Server sends detailed contact report with subject node's route
Actual Result	1. Target route was not found in database and subject node was requested for custom route 2. Subject node created custom route to target and sent to server 3. Server included custom route in detailed contact report and all nodes sync and updated
Pass/Fail	Pass

6.2 Client Application Tests

Test Case ID	1
Unit to Test	Login
Assumptions	1. User not logged in 2. User exists 3. Login page opened
Test Data	1. Username 2. Password
Steps to be Executed	1. Visit login page 2. Enter username 3. Enter password 4. Click login button
Expected Result	Reload the application page with user logged in
Actual Result	Application page is reloaded with user logged in
Pass/Fail	Pass

Test Case ID	2
Unit to Test	Logout
Assumptions	1. User logged in 2. User exists 3. Application running with user logged in status
Test Data	User session information
Steps to be Executed	1. Click on logout button
Expected Result	Reload the application page with user logged out
Actual Result	Application page is reloaded with user logged out
Pass/Fail	Pass

Test Case ID	3
Unit to Test	Map Initialization
Assumptions	1. Application is running 2. User logged in
Test Data	1. Own position from GPS 2. Availability of maps in cache 3. Place marker of own position on map
Steps to be Executed	1. Application started by user 2. User logs in 3. Application checks for own position Grid Reference (GR) from GPS 4. Based on own GR, application checks for corresponding map tiles in cache 5. Load the map tiles from cache 6. Place marker for own position on map
Expected Result	Map is loaded from cache with own position GR marker placed on it

Actual Result	Map loaded and own position GR marker visible on it
Pass/Fail	Pass

Test Case ID	4
Unit to Test	Synchronization
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application running
Test Data	1. Synchronization of data between server and node 2. Node updated 3. Place marker of respective own position on server and nodes respectively 4. Place markers of all nodes on map on node 5. Response time of synchronization
Steps to be Executed	1. Server application started 2. Client application(s) started 3. Sync button clicked on the node 5. Load the map tiles from cache of respective applications 6. Place marker(s) for own position on maps of respective application(s)
Expected Result	Synchronization is successful, map grid updated on the node and server with each having place markers of all nodes and server
Actual Result	Synchronization successful, map grid updated with own position of all the elements in the network on the node
Pass/Fail	Pass

Test Case ID	5
Unit to Test	Initial Contact Report / Artillery Target Message
Assumptions	1. Client-Server network established 2. Server application is running 3. Client application running 4. Node in sync with server
Test Data	1. Sending of initial contact report to server by a client 2. Reception of initial contact report at server 3. Server acknowledging the report 4. Plotting of enemy elements on map on server based on sent GRs 5. Server sending enemy GRs to all other nodes
Steps to be Executed	1. Client sees an enemy and creates an initial contact report 2. Click send button to send report to server 3. Server updates the enemy position on map 4. Server send sync messages to all other nodes to update their maps with enemy GRs 5. All other nodes update their maps
Expected Result	Initial contact report is received at server, server updates its maps and all other nodes' maps and an acknowledgment is sent to the node
Actual Result	Enemy sighted by node, initial contact report was made, sent to server. Server sync with all other participating clients and updated them. Subject node was acknowledged of the report
Pass/Fail	Pass

Test Case ID	6
Unit to Test	Packet Types / Casualty Report
Assumptions	1.Transfer of packets between server and node 2. Packets bearing their type according to intended message

	3. Packet(s) successfully received at either side i.e. server or client
Test Data	1. Checking of types of data packets 2. Display of information on type of data packets 3. Triggering alarm / events on sensitive types of reports like casualty report etc. 4. Storage / update of information in database depending on type of data packets
Steps to be Executed	1. Client sends a report 2. Server accepts the report 3. Server checks the type of report 4. Server triggers the alarm, if necessary, to take earliest possible action 5. Server updates the database for corresponding record table 6. Display the alert of desired report
Expected Result	Desired report is displayed on server/client, and updated in database
Actual Result	1. Casualty report was sent by the node 2. Server triggered the alarm message 3. Server alerted the authorities to take immediate action 4. Node was informed of the action with an acknowledgment
Pass/Fail	Pass

Chapter 7: Conclusion and Future Work

7.1. Conclusion

Battlefield Information and Management System was developed keeping in mind the current requirements of Pakistan Army. Endeavors have been made to meet all the critical requirements, as of now, and subsequently extend them to incorporate other elements of our armed forces. Although it is the first prototype of its kind, we are hopeful that BIMS will act as first step to achieving the capability of Network Centric Warfare.

BIMS has been designed and developed, so far, to accommodate an Armored Regiment of Pakistan Army with an extension to include Artillery Observer participating in an operation. It fulfills the basic needs of an operational situation under consideration. Fruitful efforts have been underway to make BIMS more mature in terms of robustness, reliability, availability and security. Overall, BIMS will be a novel product to dictate such efforts in future.

7.2. Future Enhancements

Planning has been made for future enhancements in the following areas:

7.2.1 BIMS App for Android Platform

Android has been an emerging trend in latest electronic devices like smart phones, tablets, PCs etc. The Android Software Development Kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, Windows XP or later; for the moment one cannot develop Android software on Android itself. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

7.2.2 Extending BIMS Capabilities

To achieve the dream of Network Centric Warfare, all the three elements of armed forces have to be included in this network. BIMS will be extended to incorporate relationships to these elements and further implement a hierarchy of actions to achieve collaborated efforts in areas of operation.

7.2.3 Enhancement of Security Protocols

Proprietary encryption algorithms are based on set rules and regulations. Therefore they are prone to attacks and often found vulnerable to them. Development of own military grade algorithms will guarantee the secured communication of SDR channels and will further enhance BIMS's capabilities in terms of reliability and security.

APPENDIX A: GLOSSARY

BIMS	-	Battlefield Information and Management System
GIS	-	Geographical Information System
C3	-	Command and Control Console
SDR	-	Software Defined Radio
PHP	-	Hypertext Preprocessor (Programming Language)
MySQL	-	My. Structured Query Language
Windows CE	-	Windows Compact Edition / Embedded Compact
SDK	-	Software Development Kit
Ammo	-	Ammunition
POL	-	Petroleum, Oil & Lubricants
API	-	Application Programming Interface

BIBLIOGRAPHY

1. http://en.wikipedia.org/wiki/Blue_Force_Tracking
2. http://en.wikipedia.org/wiki/Windows_CE
3. https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
4. http://en.wikipedia.org/wiki/Google_Maps#Google_Maps_API
5. https://en.wikipedia.org/wiki/PHP
6. https://en.wikipedia.org/wiki/Mysql