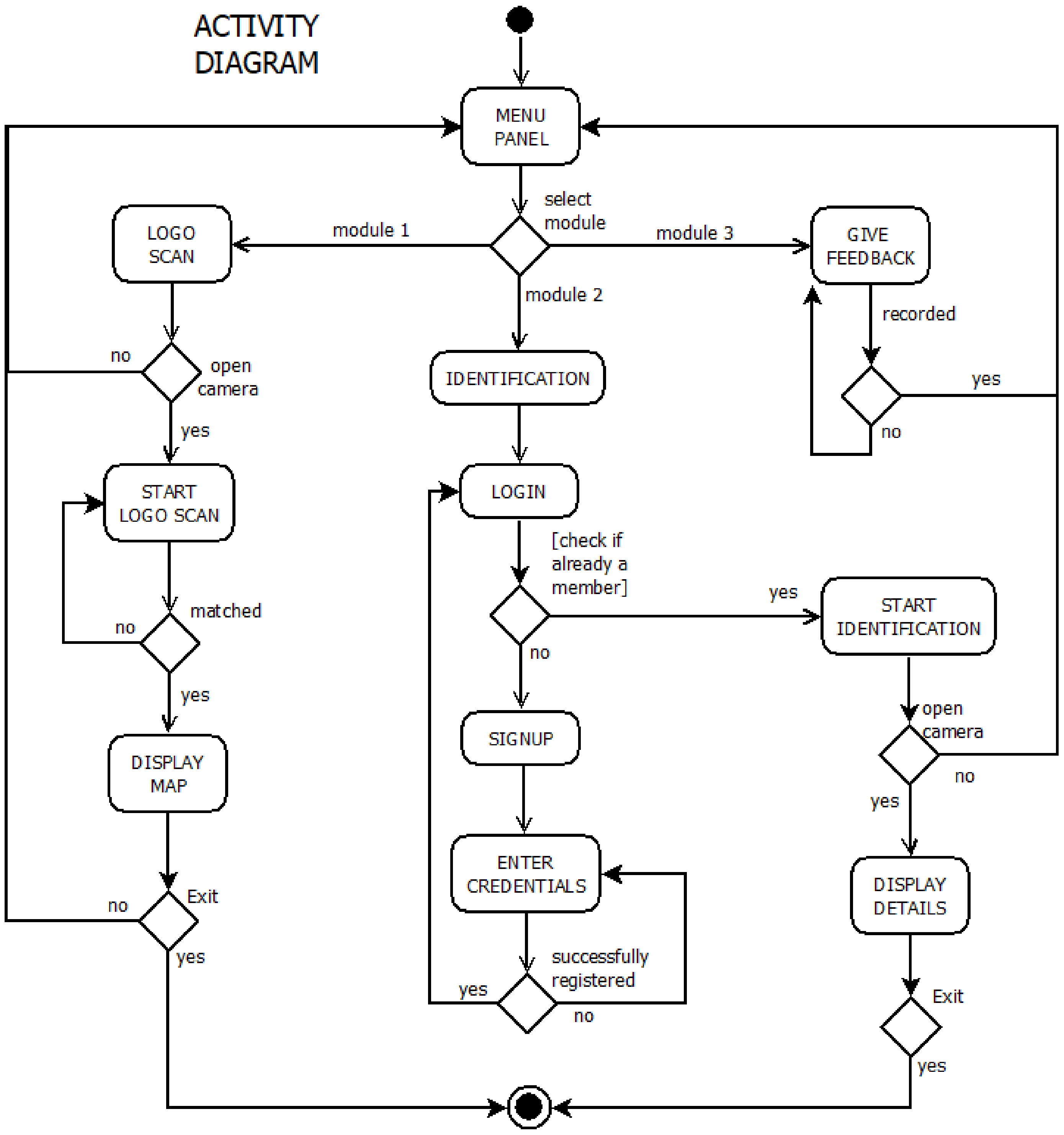
# ASSIGNMENT 1

## ACTIVITY DIAGRAM



With the initialization of initial node, Menu Panel will be opened. Menu Panel consists of 3 buttons which corresponds to respective module. On selection of component, it will load corresponding module. Project includes 3 modules named Logo Scan, Identification, and Feedback.

Logo Scan and Feedback are independent. Anybody having application can use these modules. While for using Identification user must have a valid account.

If user selects Logo Scan, application will ask required permission to on camera and give instruction to user to scan required logo to get details. On successful pointing on SGSITS logo, an augmented real virtual object (SGSITS campus map animated object) will appear. On clicking on this augmented real object, application will take user on entire campus tour of SGSITS. From this point user can easily come back and use other modules.

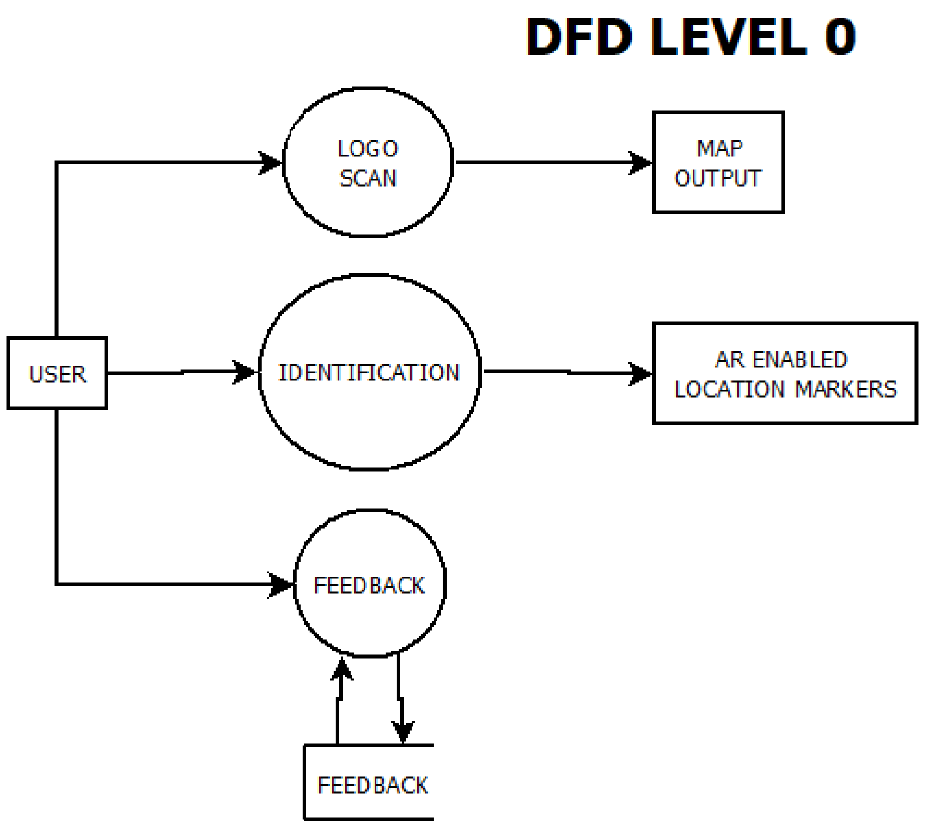
If user selects Identification, application will ask first to sign in to its valid account. If the user is not registered then there will be an option of sign up. On successful login, application will take permission to on camera to start identification of buildings. If user is near any authenticated building, its detail in augmented real object (virtual image with animation) will appear.

If user selects feedback, application will ask few details including valid Email, valid Message for application. On giving all details successfully, user’s response will be saved in admin database.

# ASSIGNMENT 2

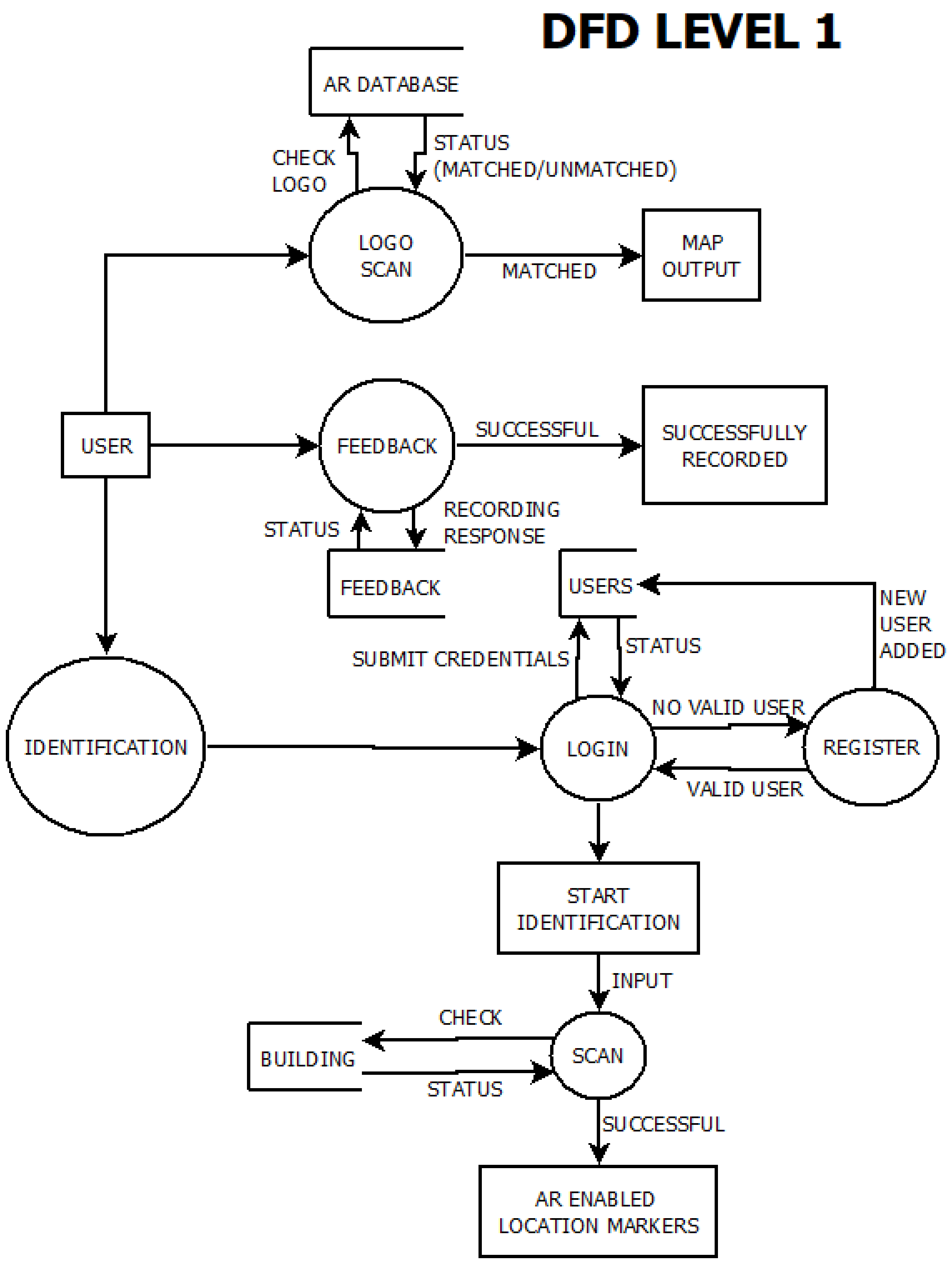
## DATA FLOW DIAGRAM

### LEVEL 0

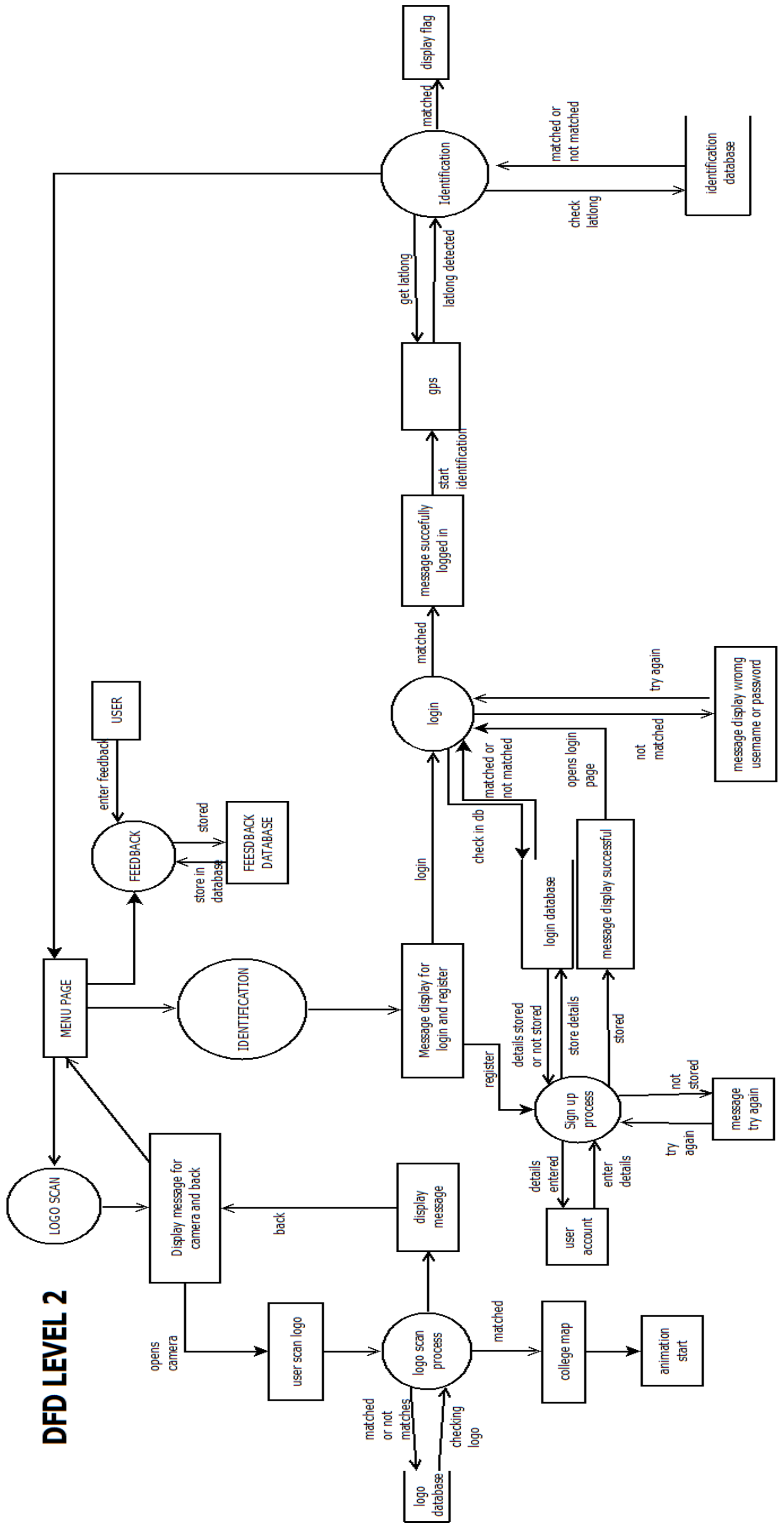


Data flow in the application is with the transition of module from one state to another. As the above diagram depicts, user can chose any one of the process amongst the three specified. Map output and AR enabled location markers are from application database which will pop out on satisfaction of correct condition. While feedback gets submitted to application database hosted on godaddy server.

## LEVEL 1


DFD LEVEL 1
AR DATABASE
CHECK LOGO
STATUS (MATCHED/UNMATCHED)
LOGO SCAN
MATCHED
MAP OUTPUT
USER
FEEDBACK
SUCCESSFUL
SUCCESSFULLY RECORDED
STATUS
RECORDING RESPONSE
FEEDBACK
USERS
NEW USER ADDED
SUBMIT CREDENTIALS
STATUS
IDENTIFICATION
LOGIN
NO VALID USER
REGISTER
VALID USER
START IDENTIFICATION
INPUT
CHECK
BUILDING
SCAN
STATUS
SUCCESSFUL
AR ENABLED LOCATION MARKERS

## LEVEL 2


DFD LEVEL 2
LOGO SCAN
MENU PAGE
IDENTIFICATION
FEEDBACK
USER
enter feedback
stored
store in database
FEESDBACK DATABASE
Display message for camera and back
opens camera
user scan logo
logo scan process
matched or not matches
checking logo
logo database
display message
back
matched
college map
animation start
Message display for login and register
register
login
Sign up process
details entered
enter details
user account
details stored or not stored
store details
stored
not stored
try again
message try again
login database
message display successful
check in db
matched or not matched
opens login page
not matched
try again
message display wromg username or password
matched
message succefully logged in
start identification
gps
get latlong
latlong detected
Identification
check latlong
matched or not matched
identification database
matched
display flag

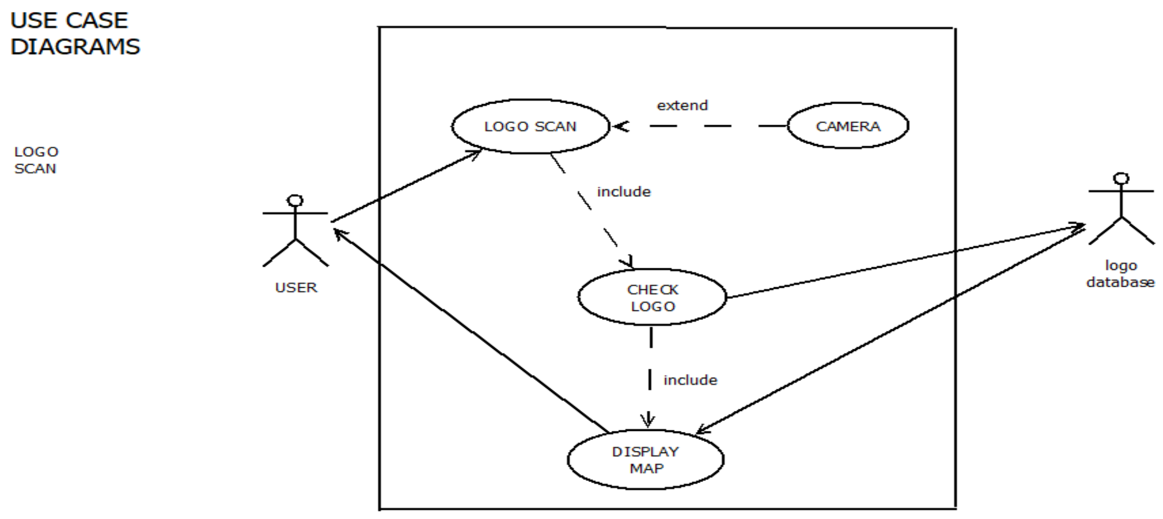
On selection of logo scan module user will start communicating with logo scan process. This process will take input image from user and check it with SGSITS logo in application database, after matching it will receive status(matched or unmatched) from database. If logo matched then there will be appearance of graphical output (augmented real object), but if logo unmatched then it will move to non matched state and give output to user.

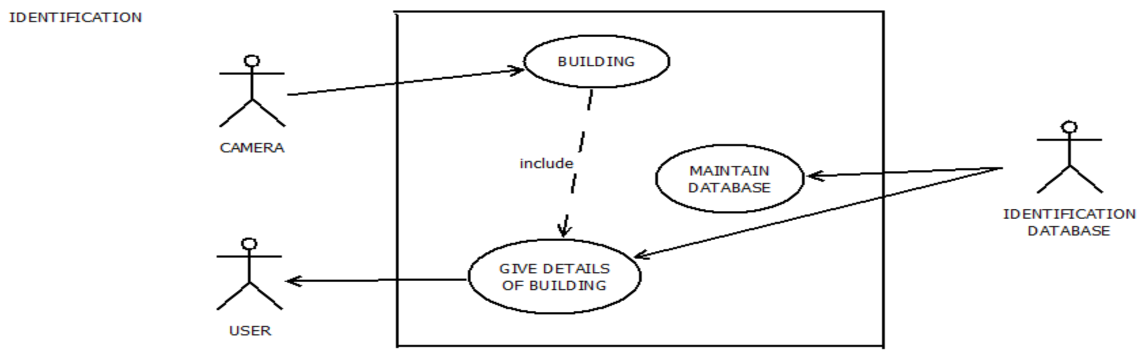
On selection of Feedback module, user submit its response to feedback process. This process will then send the response to database. After successful recording of response to database, this process will move to another state of successful recording feedback.

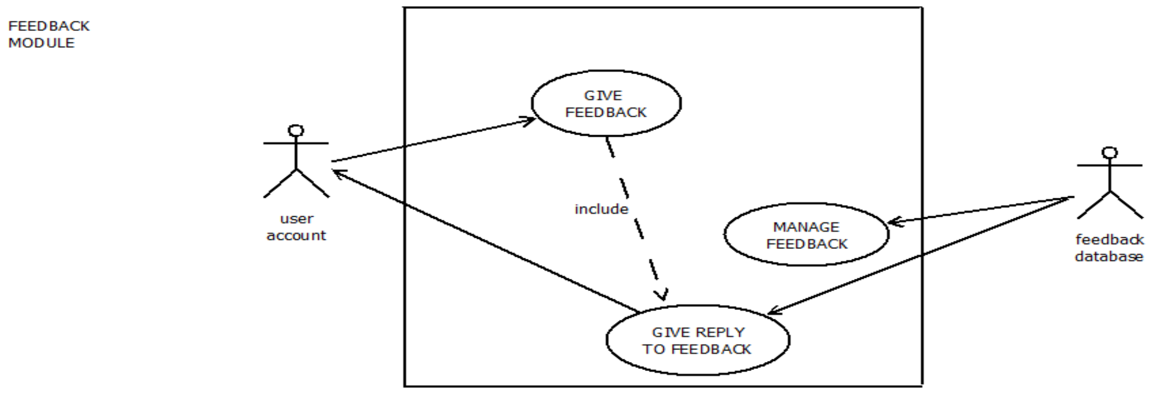
On selection of Identification module, user has to first login with valid account, if user does not have any valid account then it has to register with application database. On successful login user will be redirected to start identification where on pointing device to any building will submit images to database and checks for building identification. On receiving successful status, graphical details (augmented real object) will appear, else will give error.

# ASSIGNMENT 3

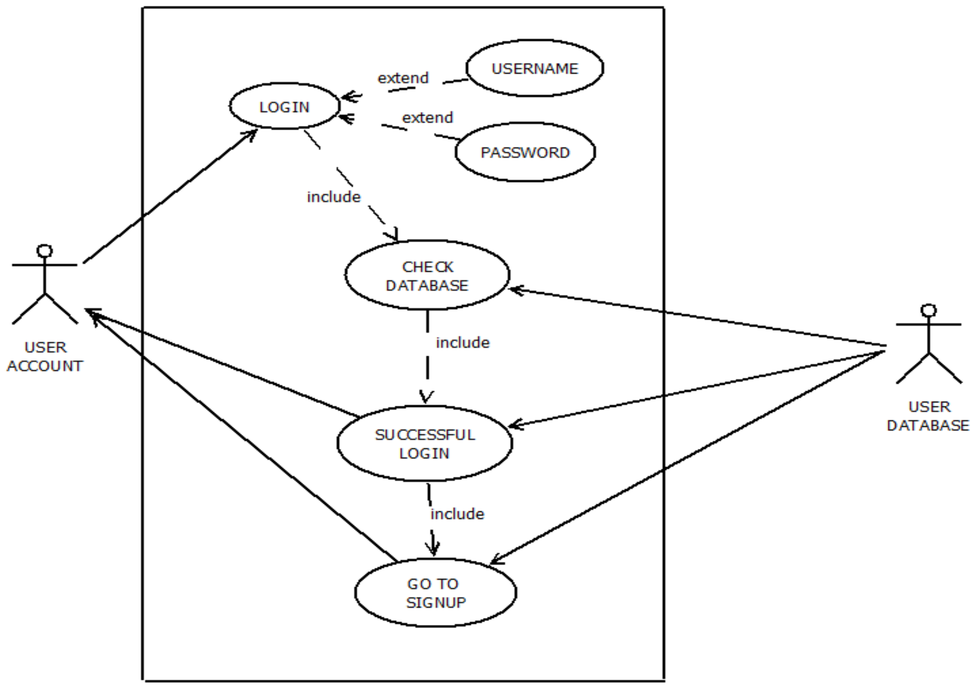
## USE CASE DIAGRAM



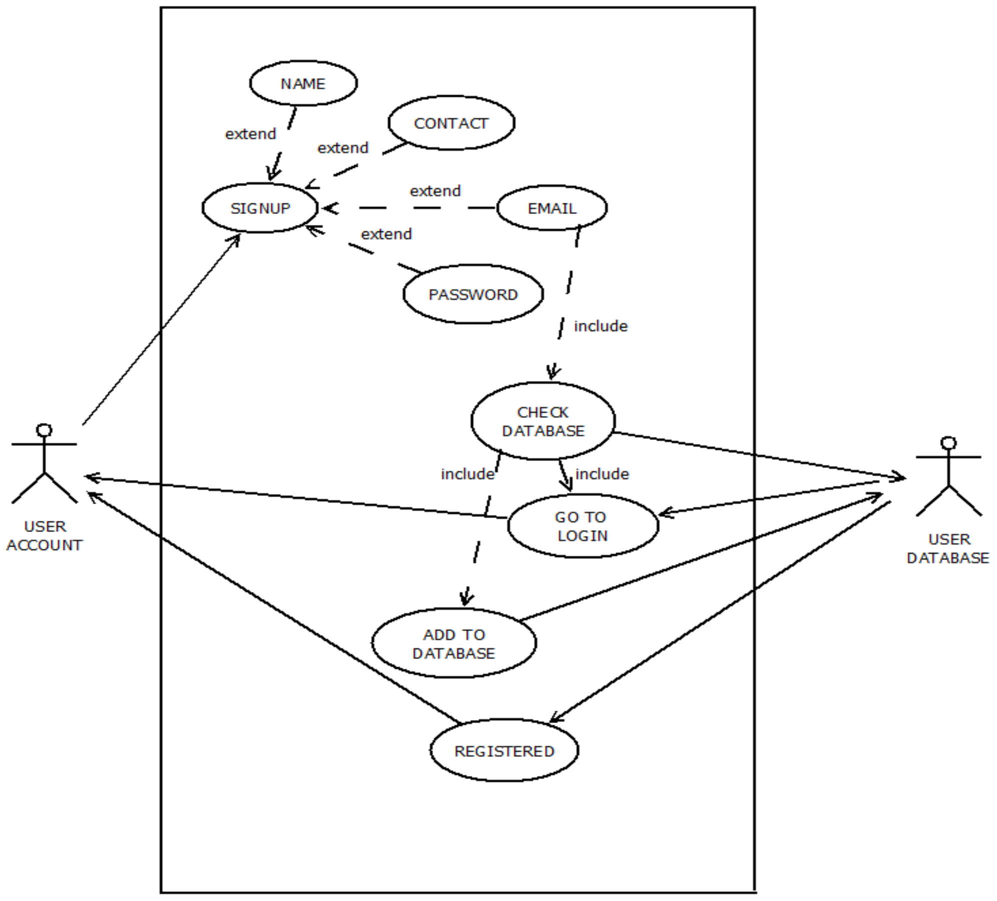




LOGIN
MODULE



SIGNUP
MODULE



We’ve made the use cases for each module in the application. A brief description of each use-case is as follows:

1. LOGO SCAN –
   In this use case, we have two actors user and logo database which communicate with the system. The users will instantiate the logo scan process which will begin only with camera.
   After the user scans the logo, the scanned is verified from the logo database, whether the input scanned is correct or not.
   If the input is successfully found in the database, the user is given a output in form of traversal through the map of the college

2. IDENTIFICATION-
   Here, we have three actors namely USER,CAMERA and IDENTIFIATION DATABASE.
   The camera first point towards the building whose latitudes and longitudes are scanned and verified with the identification database, and thus, if successfully verified, the output is given to the user.

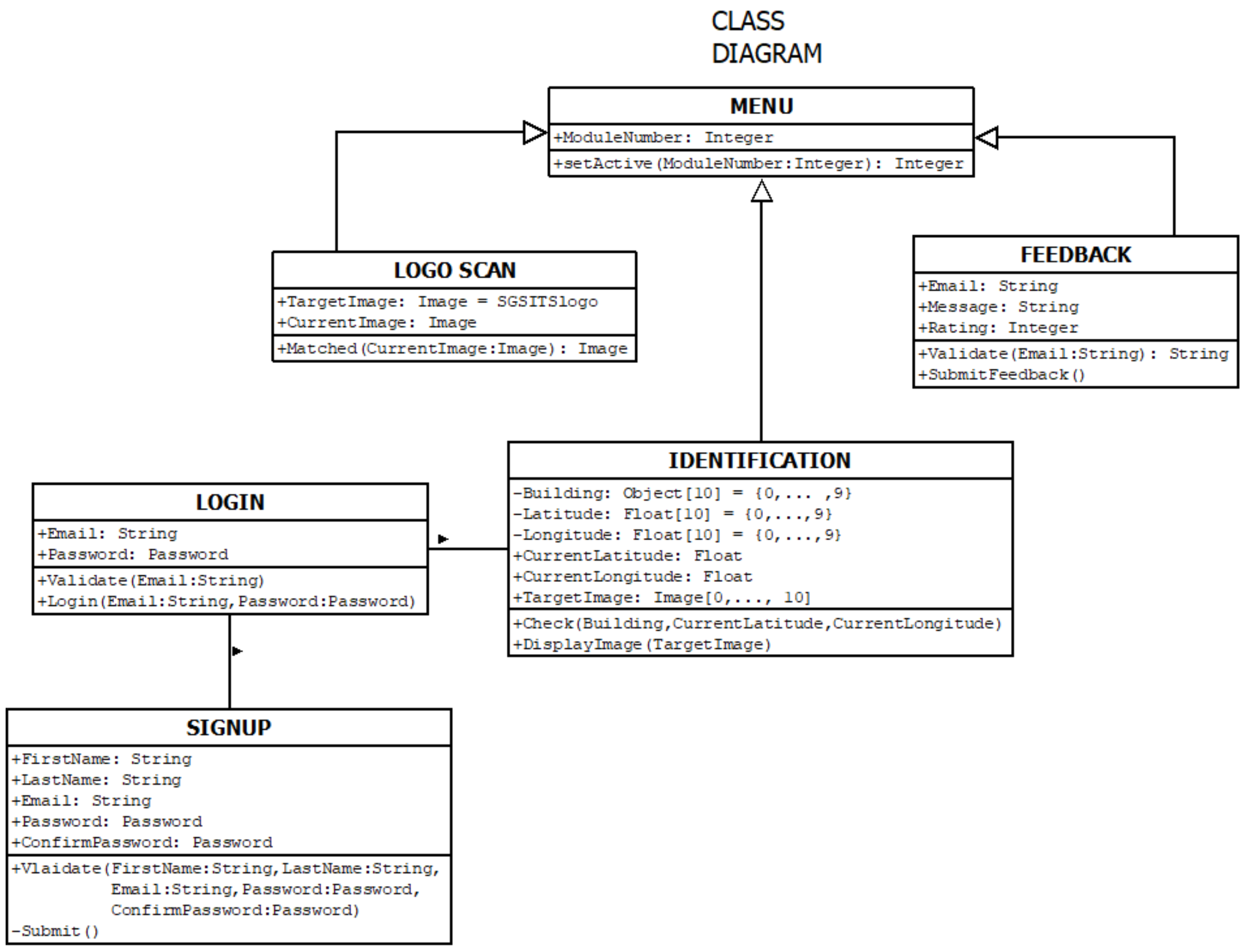
3. FEEDBACK MODULE-
   The first actor, User give its feedback, which is then recorded into the feedback database.
   The actor, feedback database along with managing the feedbacks which are fed into it, also generates a reply to feedback which is given as output to the user.

4. LOGIN-
   The actor, user account, has to enter his/her email(treated as primary key) and password to login. The credentials entered are then checked through the actor, USERDATABSE. If the user database check is successful then user is given a message of ‘successful login’ else he/she is redirected to the signup module.

5. SIGNUP-
   In the signup module, the actor user has to enter its name, contact, email and password to continue. The primary key, email is first cross-checked into the user database to verify that no similar email exists. If it is successfully verified then the user is added into the user database, and the user is given a message as ‘registered.’ If the email already exists, the user is directed towards the login page.

# ASSIGNMENT 4

## CLASS DIAGRAM


CLASS DIAGRAM
MENU
+ModuleNumber: Integer
+setActive(ModuleNumber:Integer): Integer
LOGO SCAN
+TargetImage: Image = SGSITSlogo
+CurrentImage: Image
+Matched(CurrentImage:Image): Image
FEEDBACK
+Email: String
+Message: String
+Rating: Integer
+Validate(Email:String): String
+SubmitFeedback()
IDENTIFICATION
-Building: Object[10] = {0,... ,9}
-Latitude: Float[10] = {0,...,9}
-Longitude: Float[10] = {0,...,9}
+CurrentLatitude: Float
+CurrentLongitude: Float
+TargetImage: Image[0,..., 10]
+Check(Building,CurrentLatitude,CurrentLongitude)
+DisplayImage(TargetImage)
LOGIN
+Email: String
+Password: Password
+Validate(Email:String)
+Login(Email:String,Password:Password)
SIGNUP
+FirstName: String
+LastName: String
+Email: String
+Password: Password
+ConfirmPassword: Password
+Vlaidate(FirstName:String,LastName:String, Email:String,Password:Password, ConfirmPassword:Password)
-Submit()

Project includes the use of major 6 classes named Menu, Logo Scan, Feedback, Identification, Login and Signup.

- MENU: Menu panel comprises of Menu class, which has setActive method which takes integer as parameter which is the index of the module and hence loads corresponding module.

- LOGO SCAN: Logo scanning module comprises of Logo Scan class which inherits menu class, this class has matched method which checks for similarity in input logo and logo already present in database.

- FEEDBACK: Feedback module comprises of feedback class which inherits menu class, this class has 2 methods named validate-which checks for valid user email, valid message, valid rating and submitFeedback –which stores the response of user to admin database.

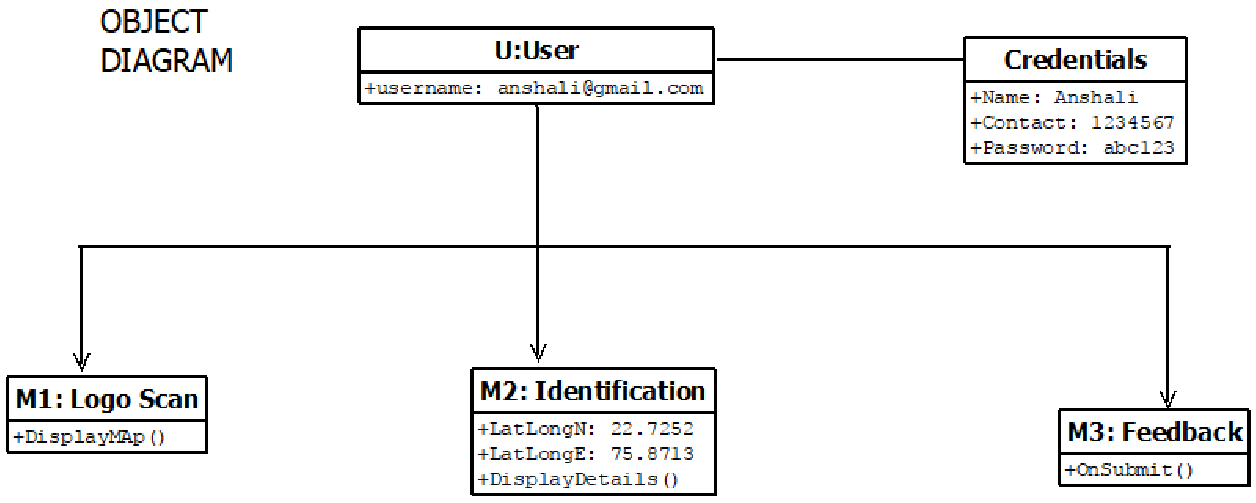
- IDENTIFICATION: Identification module comprises of identification class which inherits menu class, this class has 2 methods named check-checks for current latitude and longitude with the existing latitudes and longitudes of the appropriate distance between two and displayImage-will display appropriate target image according to the distance calculated.

- LOGIN: Login class is associated with Identification class, as to access members and methods of identification class user should have registered with application database. This class has 2 methods named validate-which checks for valid Email and login-which check for correct credentials with the database and allow access.

- SIGNUP: Signup class is associated with login class, if the user is not registered with application database, this class will ask for credentials needed for creating an account. It has 2 methods named validate-which checks for valid name, email, password etc and Submit-which stores credentials into application database and creates an account.

# ASSIGNMENT 5

## OBJECT DIAGRAM



The following explains the terms used in the diagram:

- U: User- At the particular instance, the user with email (used as primary key) anshali@gmail.com is logged into the application.

- Credentials- The user which is logged in has the Credentials as Anshali, 123456, abc123 as the name, contact and password respectively.

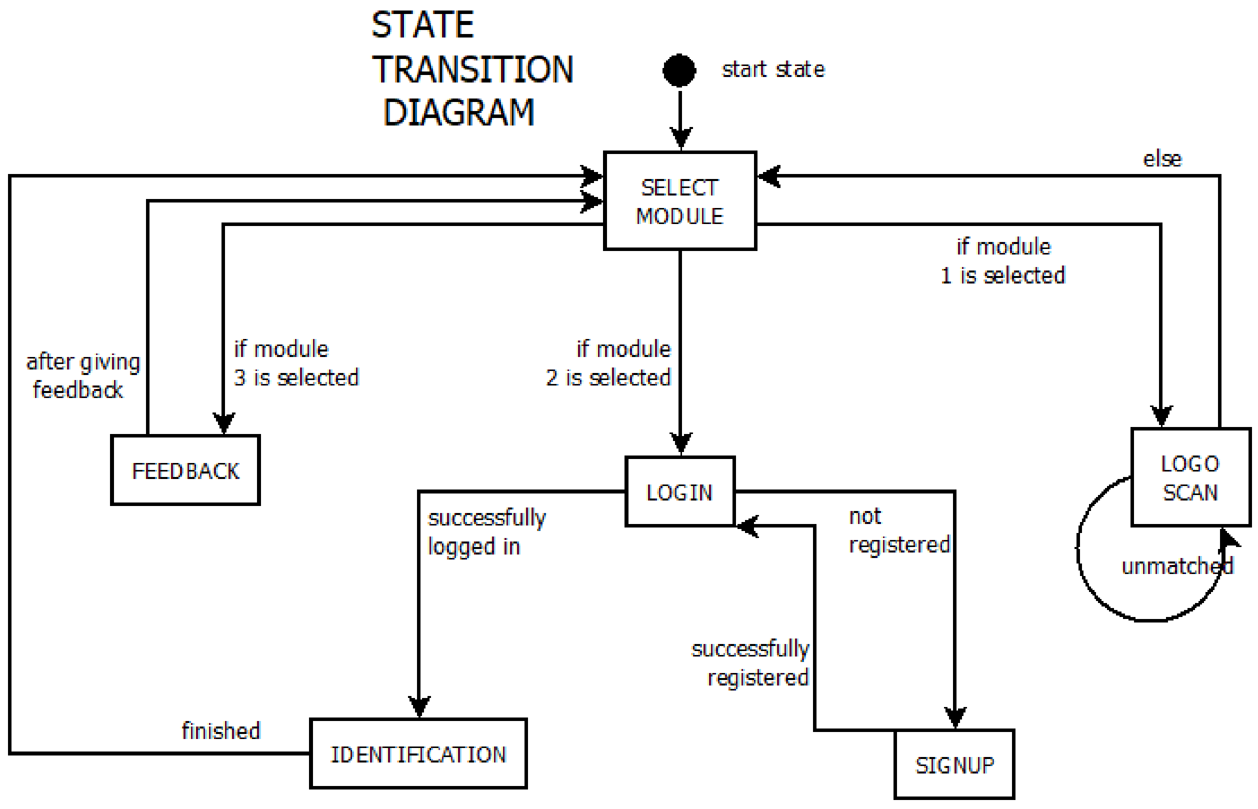
- M1: Logo Scan- the first module is the logo scan module where it displays the requisite map on correct logo scan.

- M2: Identification- The second module is the Identification module, in which the user anshali have the present location as latitude- 22.7252 and longitude-75.8713 and it's displaying the details of the present location where the user is.

- M3: Feedback- The feedback of the user is taken through the third module where the method onSubmit feeds the data into the database whenever 'submit' button is clicked.

# <u>ASSIGNMENT 6</u>

## STATE TRANSITION DIAGRAM



State transition diagram includes all relationships amongst various states. This Project includes 6 major states named select module, feedback, login, logo scan, identification, signup. Explanations of various states are as follows:

After start application will land up on Select Module State. From Select module state there can be 3 transitions to feedback state, logo scan state and login state to use identification functionality.

- If Feedback stated is selected, then application will ask for valid email and message .After successfully recording feedback, there is one transition from feedback state to again select module state.

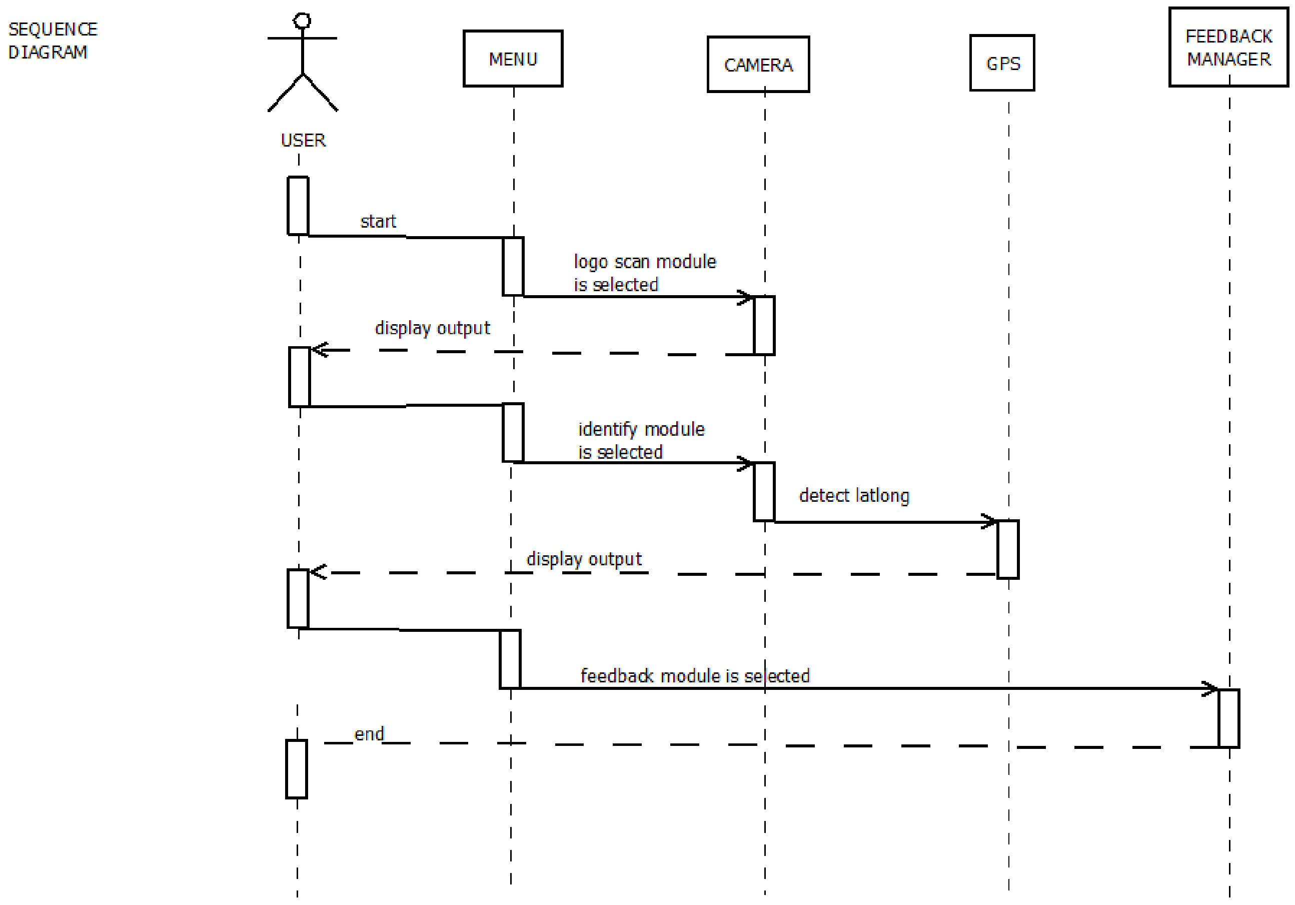
- If Logo Scan state is selected, application will on camera and asks for valid input. On logo scan state, if logo is unmatched, it will remain on same state and if user wants to end this app will move back to menu to select other module.

- If Identification is selected, identification first we move to login state after select module state. On unsuccessful login we move to signup state to create an account. After Successful login we move to Identification state. After Identification we can move to select module state again to experience other modules.

These are various transitions of application from various states.

# ASSIGNMENT 7

## SEQUENCE DIAGRAM



Sequence diagram depicts the sequence in which different modules or functionalities are accessed by user. A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

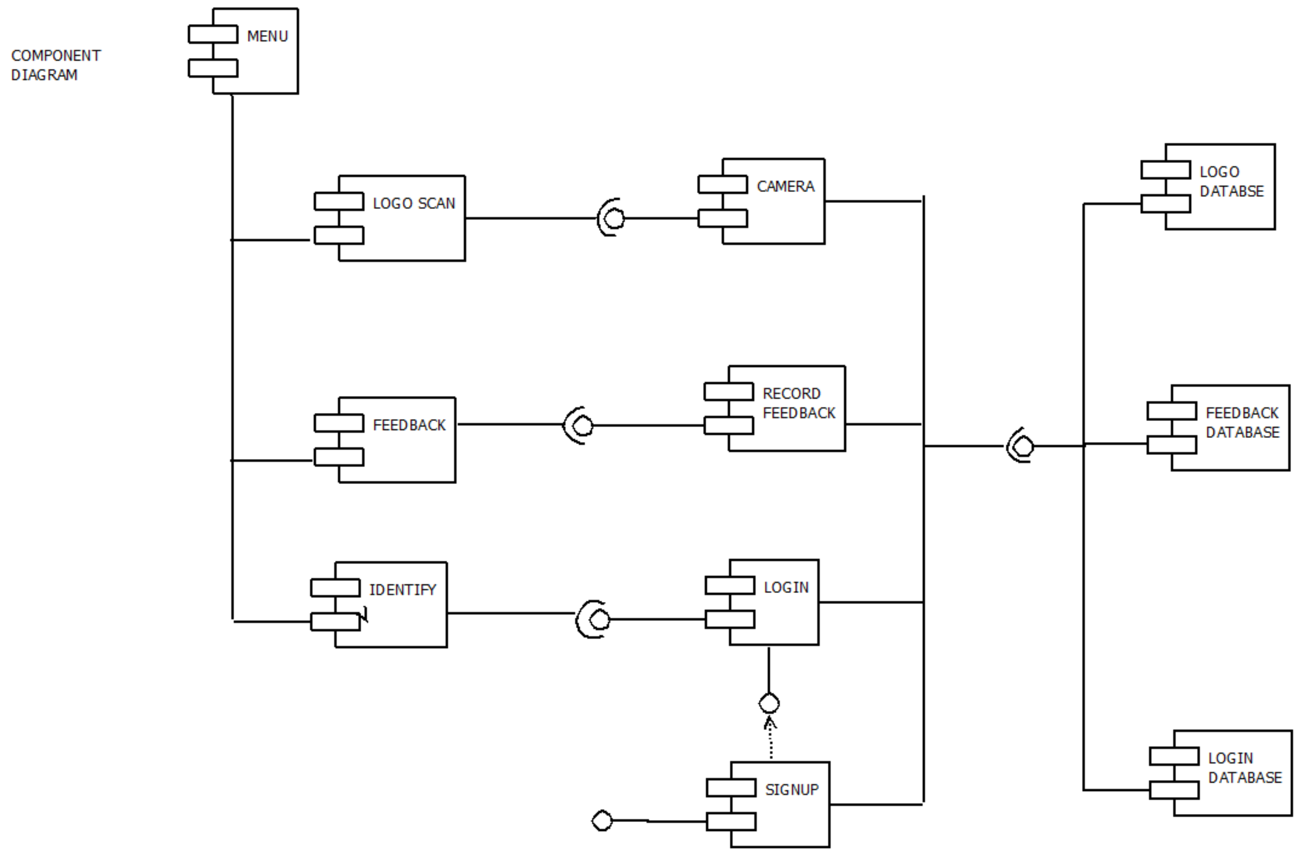
- In the application user enters, it first encounters menu, from which it selects one amongst the three functionalities. If selects logo scan then after menu, camera opens and scanning is done.

- On selection of identification, after camera with the help of GPS of device and various buildings calculations done and accordingly results get displayed.

- On selection of feedback, after GPS, feedback module gets loaded where user gets direct connection with the developer team.

# ASSIGNMENT 8

## COMPONENT DIAGRAM



Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams. It does not describe the functionality of the system but it describes the components used to make those functionalities.

This application is developed in Unity3D with Vuforia SDK for Augmented reality. If user want to experience AR just by sitting on its place, Logo scan module is used, which used phone camera for scanning of logo and Vuforia’s AR camera for showing output of graphical object. Unity’s local player is used to run animation.

Another application of augmented reality used in this application is Building recognition. For this phone camera is used to bring out graphical augmented real object. Phone GPS is used for recognition of location and other required calculations are made. To experience this module user has to register with application, hence godaddy’s hosting server is used to manage user database and proper login communication.

To connect directly with the developers, application gives feedback system where any user with application can give proper email and submit any message, query or anything with the team. Feedback also gets recorded on godaddy’s hosting server.

# ASSIGNMENT 9

## 1. Application of Object Oriented Analysis and Design

### 1.1 Applying OOAD in CRYPTANALYSIS :

#### 1.1.1 Requirement :

The requirement of our problem can be stated briefly as : devise a system that, given a cryptogram, transforms it back to its original plaintext, assuming that only a simple substitution cipher was employed.

#### 1.1.2 Analysis :

Cryptanalysis, the process of transforming cipher text back to plaintext.
A blackboard framework satisfies our definition for a framework, where we can define the architecture in terms of a set of classes and mechanisms that
Describe how instances of those classes collaborate.
The basic ideology of blackboard model is - "the purpose of the blackboard is to hold computational and solution-state data needed by and produced by the knowledge sources. The blackboard consists of objects from the solution space. The objects on the blackboard are hierarchically organized into levels
of analysis. The objects and their properties define the vocabulary of the solution space".
The knowledge sources for our system are:
Common prefixes Common word beginnings such as re, ant, and un
• Common suffixes Common word endings such as ly, ing, es, and ed
• Consonants No vowel letters
• Direct substitution Hints given as part of the problem statement
• Double letters Common double letters, such as tt, ll, and ss
• Letter frequency Probability of the appearance of each letter
• Legal strings Legal and illegal combination if letters, such as quand zg, respective
• Pattern matching Words that match a specified pattern of letters
• Sentence structure Grammar, including the meanings of noun and verb, phrases
• Small words possible matches for one-, two-, three-, and four-letter words
• Solved Whether or not the problem is solved, or if no
Further progress can be made
• Vowels Nonconsonant letters

• Word structures the location of vowels and the common structure of nouns, verbs, adjectives, adverbs, articles, conjunctives, and so on.

From an object-oriented perspective, each of these thirteen knowledge sources represents a candidate class in our architecture: each instance embodies some state (its knowledge), each exhibits certain class-specific behavior (a suffix knowledge source can react to words suspected of having a common ending), and each is uniquely identifiable. We may also arrange these knowledge sources in a hierarchy. Specifically, some knowledge sources operate upon sentences, others upon letters, still others on contiguous groups of letters, and the lowest-level ones on individual letters. Indeed, this hierarchy reflects the objects that may appear on the blackboard: sentences, words, strings of letters, and letters.

### 1.1.3 Design:

Blackboard Objects the objects that appear on a blackboard exist in a structural hierarchy that parallels the different levels of abstraction of our knowledge sources. Thus, we have the following three classes:

- **Sentence** - A complete cryptogram
- **Word** - A single word in the cryptogram
- **Cipher Letter** - A single letter of a word

Knowledge sources must also share knowledge about the assumptions each makes, so we include the following class of blackboard objects:

- **Assumption** - An assumption made by a knowledge source

Finally, it is important to know what plaintext and cipher text letters in the alphabet have been used in assumptions made by the knowledge sources, so we include the following class:

- Alphabet - The plaintext alphabet, the cipher text alphabet, and the mapping between.

Looking at Blackboard Object class from its outside view,we may define two applicable operations:

- register-Add the object to the blackboard
- resign- Remove the object from the blackboard

We also define classes Dependent, Affirmation and Assertion, to classify whether input is assumed or asserted or is dependent on some other fields.

### 1.1.4 Evolution and Maintenance :

As we have defined the key abstractions for our domain, we may continue by putting them together to form a complete application. We will proceed by implementing and testing a vertical slice through the architecture, and then by completing the system one mechanism at a time. We will proceed as follows:

1. Integrating the Topmost Objects(Blackboard Object and knowledge Sources)
2. Adding New Knowledge Sources

We make our design susceptible to improvements by following-
1. Adding New Functionality –
We desire our application to be introspective: it should keep track of when knowledge sources were activated, what assumptions were made and why, and so on, so that we can later question it, for example, about why it made an assumption, how it arrived at another assumption, and when a particular knowledge source was activated.
2. Changing the Requirements –
Some new requirements that can come in future can be-
• The ability to decipher languages other than English
• The ability to decipher using transposition ciphers as well as single-substitution ciphers
• The ability to learn from experience

Once we have a stable implementation in place,many new requirements can be incorporated with minimal change to our design.

## 1.2 Applying OOAD in WEATHER MONITORING SYSTEM

### 1.2.1 Requirements –

This system shall provide automatic monitoring of various weather conditions. Specifically, it must measure:

• Wind speed and direction
•Temperature
• Barometric pressure
• Humidity

The system shall also provide the following derived measurements:

• Wind chill

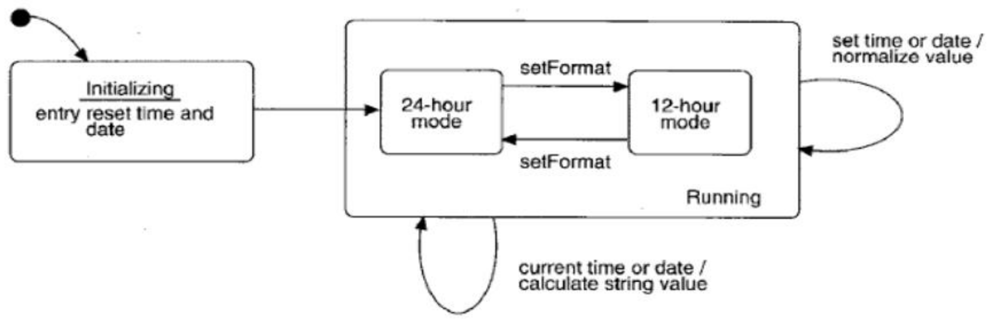
• Dew point temperature
• Temperature trend
• Barometric pressure trend
The system shall have a means of determining the current time and date, so that can report the highest and lowest values of any of the four primary measurements during the previous 24 hour period.

The system shall have a display that continuously indicates all eight primary and derived measurements, as well as the current time and date. Through the use of a keypad, the user may direct the system to display the 24-hour high or low value of any one primary measurement, together with the time of the reported value. The system shall allow the user to calibrate its sensors against known values, and to set the current time and date.

### 1.2.2 Analysis –

- **Defining the Boundaries of the Problem –**
  We begin our analysis by considering the hardware on which our software must execute. This is inherently a problem of systems analysis, involving manufacturability and cost issues. Then we make strategic assumptions like -
  - We will use a single-board computer(SBC) with a 486-class processor 75.
  - Time and date are supplied by an on-board clock, accessible via memory-mapped I/O.
  - Temperature, barometric pressure, and humidity are measured by on-board circuits (with remote sensors), also accessible via memory-mapped I/O.

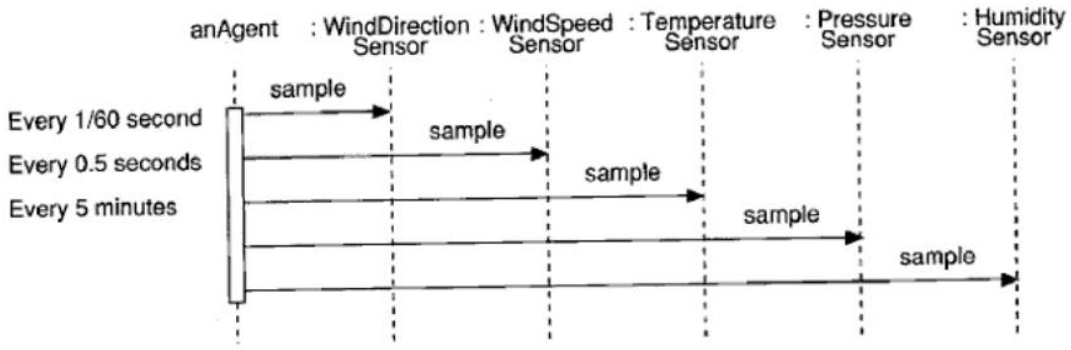
Then, we define various classes associated with each functionality of the required system. For example, the outside view of the time-date class can be understood by the diagram :


Initializing
entry reset time and date
24-hour mode
12-hour mode
setFormat
setFormat
Running
set time or date / normalize value
current time or date / calculate string value

- **Scenarios-**

  Now that we have established the abstractions atthe boundaries of our system, we continue our analysis by studying several scenarios of its use. Monitoring basic weather measurements is the principle function point of the weather monitoring system. One of our system constraints is that we cannot take measurements any faster than 60 times a second. Our analysis suggests that the following sampling rates are sufficient to capture changing conditions:

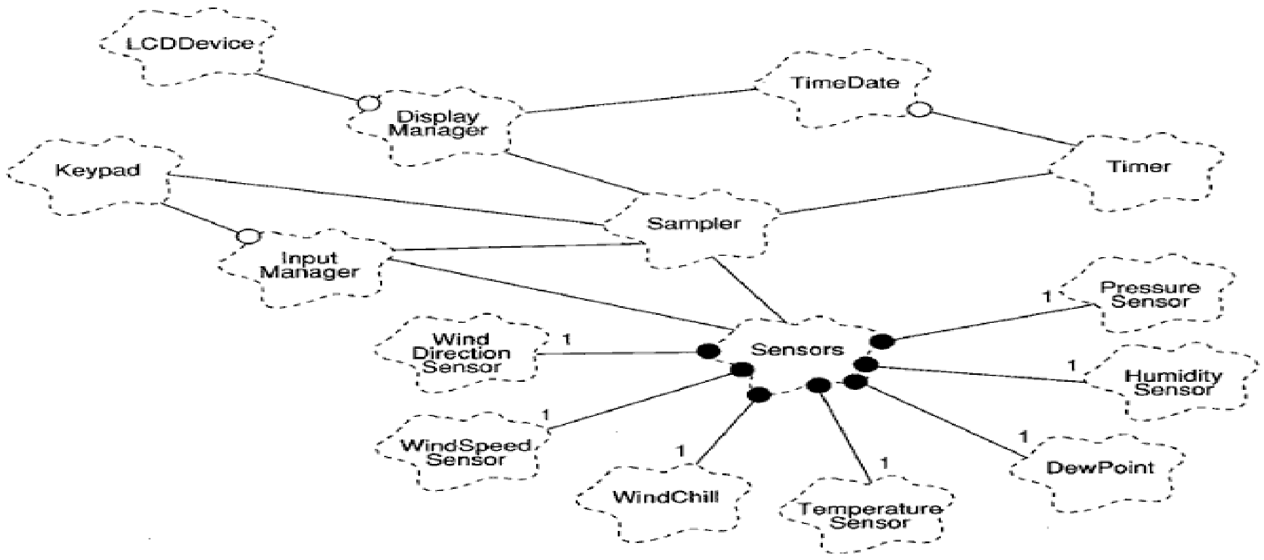
  - Every 0.1 second  wind direction
  - Every 0.5 seconds  wind speed
  - Every 5 minutes  temperature, barometric pressure, and humidity



Scenario for Monitoring Basic Measurements

### 1.2.3 Design :

- **Architectural framework :**
  In data acquisition and process-control domains, there are many possible architectural patterns we might follow, but the two most common alternatives involve either the synchronization of autonomous actors or time-frame-based processing.
  In the first pattern, our architecture encompasses a number of relatively independent objects, each of which serves as a thread of control. For example, we might invent several new sensor objects that build upon more primitive hardware/software abstractions, with each such object responsible for taking its own sample and reporting back to some central agent that processes these samples.
  The second model takes time and divides it into several (usually fixed-length) frames, which we further divide into subframes, each of which encompasses some functional behavior. The activity from one frame to another may be different. For example, we might sample the wind direction every 10 frames, but sample the wind speed is only every 30 frames.The primary merit of this architectural pattern is that we can more rigorously control the order of events.
  The following diagram deciphers the second model:



### 1.2.4 Evolution and Maintenance-

We start this process by proposing a sequence of releases, each of which builds upon the previous release:

- Develop a minimal functionality release, which monitors just one sensor.
- Complete the sensor hierarchy.
- Complete the classes responsible for managing the display.
- Complete the classes responsible for managing the user interface.

The complete implementation ofthis basic weather monitoring system is of modest size,encompassing only about 20 classes. If we want to add a new functionality, say that users want to measure rainfall as well. What will the impact of adding a rain gauge?

Using the architectural view of the system as a baseline, to implement this new feature, we must

- Create a new class RainFallSensorand insert it in the proper place in the sensor class hierarchy (a RainFallSensoris a kind of HistoricalSensor).
- Update the enumeration SensorName.
- Update the DisplayManagerso that it knows how to display values of this sensor.
- Update the InputManagerso that it knows how to evaluate the newly-defined key RainFall.
- Properly add instances of this class to the system's Sensorscollection.

We must deal with a few other small tactical issues needed to graft in this new abstraction,but ultimately, we need not disrupt the system's architecture nor its key mechanisms.

## 1.3 Applying OOAD in TRAFFIC MANAGEMENT SYSTEM

**1.3.1 Requirements-** The traffic management system has two primary functions: train routing and train-systems monitoring. Related functions include traffic planning, train location tracking, traffic monitoring, collision avoidance, failure prediction, and maintenance logging.

### 1.3.2 Analysis-

A study of the requirements for the traffic management system suggests that we really have four different sub problems to solve:

- Networking

- Database

• Human/machine interface

• Real-time analog device control

The thread that ties this system together is a distributed communications network. The safe operation of this entire system depends upon the timely and reliable transmission and reception of messages.

Additionally, this system must keep track of the current locations and planned routes of many different trains simultaneously. We must keep this information current and self-consistent, even in the presence of concurrent updates and queries from around the network. This is basically a distributed database problem.

The engineering of the human/machine interfaces poses a different set of problems. Specifically, the users of this system are principally train engineers and dispatchers, none of whom are necessarily skilled in using computers. All forms of user interaction must therefore be carefully engineered to suit this domain-specific group of users.

Lastly, the traffic management system must interact with a variety of sensors and actuators. No matter what the device, the problems of sensing and controlling the environment are similar, and so should be dealt with in a consistent manner by the system.

If we do a brief domain analysis across these four problem areas, we find that there are three common high-level key abstractions:

| | |
|---|---|
| • Trains | Including locomotives and cars |
| • Tracks | Encompassing profile, grade, and wayside devices |
| • Plans | Including schedules, orders, and clearances. |

Every train has a current location on the tracks, and each train has exactly one active plan. Similarly, the number of trains at each point on the tracks may be zero or one; for each plan, there is exactly one train, involving many points on the tracks. Continuing, we may devise a key- mechanism for each of these four nearly independent sub problems:

• Message passing

• Train-schedule planning

• Displaying

- Sensor data acquisition

These four mechanisms form the soul of our system. They represent approaches to what we have identified as the areas of highest development risk. It is therefore essential that we deploy our best system architects here to experiment with alternative approaches and eventually settle upon a framework from which more junior developers may compose the rest of the system

### 1.3.3 Design-

1.3.3.1 Message Passing: all messages are ultimately instances of a generalized abstract class named Message, which encompasses the behavior common to all messages. Three lower level classes represent the major categories of messages, namely, train status messages, train plan messages, and wayside device messages. Each of these classes is further specialized.

1.3.3.2 Train-Schedule Planning: the concept of a train plan is central to the operation of the traffic management system. Each train has exactly one active plan, and each plan is assigned to exactly one train and may involve many different orders and locations on the track.

1.3.3.3 Displaying: Using off-the-shelf technology for our database needs helps us to focus upon the domain specific parts of our problem. We can achieve similar leverage for our display needs by using standard graphics facilities.

1.3.3.4 Sensor Data Acquisition: the traffic management system includes many different kinds of sensors. For example, sensors on each train monitor the oil temperature, fuel quantity, throttle setting, water temperature, drawbar load, and so on. Finally, if this value goes far beyond its limits, then we might need to sound some sort of alarm, and notify yet another client to take drastic action.

### 1.3.4 Evolution and Maintenance-

The module is a necessary but insufficient means of decomposition; and thus, for a problem of the size of the traffic management system, we must: focus upon a subsystem-level decomposition. Two important factors suggest that an early activity of evolution should include devising the module architecture of the traffic management system, representing its physical software structure.

The software design for very large systems must often commence before the target hardware is completed. Software design frequently takes far longer than hardware design, and in any case, trade-offs must be made against each along the way. This implies that hardware dependencies in the software must be isolated to the greatest extent possible, so that software design can proceed in the absence of a stable target environment. It also implies that the software must be designed with the idea of replaceable subsystems in mind. In a command and control system such as the traffic management system, we might wish to take advantage of new hardware technology that has matured during the development of the system's software.

Old software never dies, it just gets maintained or preserved, especially for systems as large as this one. This is the reason we still find software in production use that was developed over twenty years ago (which is absolutely ancient in software years). As more users apply the traffic management system, and as we adapt this design to new implementations, clients will discover new, unanticipated uses for existing mechanisms, creating pressure to add new functionality to the system.

For example, Track layouts and wayside equipment may change over time. The numbers of trains and their routes may change daily. The system must be designed to permit incorporation of new sensor, network, and processor technology.

# ASSIGNMENT 10

## CASE STUDY: THE NEXT GEN POS SYSTEM

A POS system is a computerized application used (in part) to record sales and handle payments; it is typically used in a retail store. It includes hardware components such as a computer and bar code scanner, and software to run the system. It interfaces to various service applications, such as a third party tax calculator and inventory control. These systems must be relatively fault-tolerant; that is, even if remote services are temporarily unavailable (such as the inventory system), they must still be capable of capturing sales and handling at least cash payments (so that the business is not crippled).

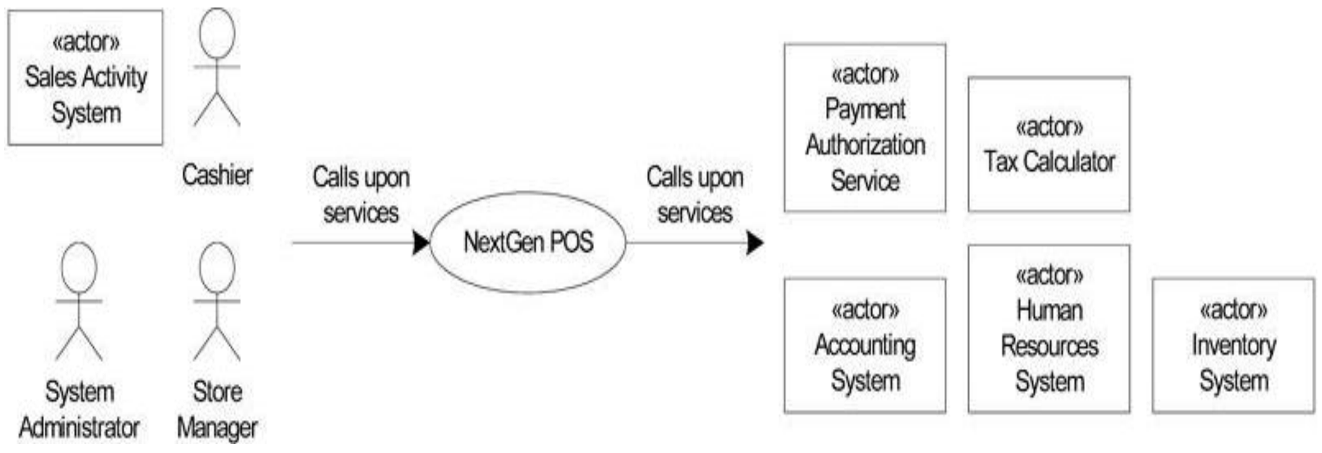
A POS system increasingly must support multiple and varied client-side terminals and interfaces. These include a thin-client Web browser terminal, a regular personal computer with something like a Java Swing graphical user interface, touch screen input, wireless PDAs, and so forth.

Furthermore, we are creating a commercial POS system that we will sell to different clients with disparate needs in terms of business rule processing. Each client will desire a unique set of logic to execute at certain predictable points in scenarios of using the system, such as when a new sale is initiated or when a new line item is added. Therefore, we will need a mechanism to provide this flexibility and customization.

### 10.1 User-Level Goals

The users (and external systems) need a system to fulfill these goals:

• Cashier: process sales, handle returns, cash in, and cash out
• System administrator: manage users, manage security, and manage system tables
• Manager: start up, shut down
• Sales activity system: analyze sales data



## 10.2 Use Case UC1: Process Sale

**Scope:** Next Gen POS application
**Level:** user goal
**Primary Actor:** Cashier

**Stakeholders and Interests:**

- Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.

- Salesperson: Wants sales commissions updated.

- Customer: Wants purchase and fast service with minimal effort. Wants easily visible display of entered items and prices. Wants proof of purchase to support returns.

- Company: Wants to accurately record transactions and satisfy customer interests. Wants to ensure that Payment Authorization Service payment receivables are recorded. Wants some fault tolerance to allow sales capture even if server components (e.g., remote credit validation) are unavailable. Wants automatic and fast update of accounting and inventory.

- Manager: Wants to be able to quickly perform override operations, and easily debug Cashier problems.

- Government Tax Agencies: Want to collect tax from every sale. May be multiple agencies, such as national, state, and county.

- Payment Authorization Service: Wants to receive digital authorization requests in the correct format and protocol. Wants to accurately account for their payables to the store.

**Preconditions:** Cashier is identified and authenticated.

**Success Guarantee (Post conditions):** Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.

**Main Success Scenario (Basic Flow):**
1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.
Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

**Extensions (Alternative Flows):**

a. At any time, Manager requests an override operation:

    1. System enters Manager-authorized mode.
    2. Manager or Cashier performs one Manager-mode operation. e.g., cash balance change, resume a suspended sale on another register, void a sale, etc.
    3. System reverts to Cashier-authorized mode.

b. At any time, System fails: To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.

    1. Cashier restarts System, logs in, and requests recovery of prior state.
    2. System reconstructs prior state.
        2a. System detects anomalies preventing recovery:
        1. System signals error to the Cashier, records the error, and enters a clean state.
        2. Cashier starts a new sale.

1a. Customer or Manager indicate to resume a suspended sale.
    1. Cashier performs resume operation, and enters the ID to retrieve the sale.
    2. System displays the state of the resumed sale, with subtotal.
        2a. Sale not found.
        1. System signals error to the Cashier.
        2. Cashier probably starts new sale and re-enters all items.

3. Cashier continues with sale (probably entering more items or handling payment).

2-4a. Customer tells Cashier they have a tax-exempt status (e.g., seniors, native peoples)

1. Cashier verifies, and then enters tax-exempt status code.
2. System records status (which it will use during tax calculations)

3a. Invalid item ID (not found in system):

1. System signals error and rejects entry.
2. Cashier responds to the error:
    - 2a. there is a human-readable item ID (e.g., a numeric UPC):
    - 1. Cashier manually enters the item ID.
    - 2. System displays description and price.
        - 2a. Invalid item ID: System signals error. Cashier tries alternate method.
    - 2b. there is no item ID, but there is a price on the tag:
    - 1. Cashier asks Manager to perform an override operation.
    - 2. Managers performs override.
    - 3. Cashier indicates manual price entry, enters price, and requests standard taxation for this amount (because there is no product information, the tax engine can't otherwise deduce how to tax it)
    - 2c. Cashier performs Find Product Help to obtain true item ID and price.
    - 2d. Otherwise, Cashier asks an employee for the true item ID or price, and does either manual ID or manual price entry (see above).

3b. There are multiple of same item category and tracking unique item identity not important (e.g., 5 packages of veggie-burgers):

1. Cashier can enter item category identifier and the quantity.

3c. Item requires manual category and price entry (such as flowers or cards with a price on them):

1. Cashier enters special manual category code, plus the price.

3-6a: Customer asks Cashier to remove (i.e., void) an item from the purchase:

This is only legal if the item value is less than the void limit for Cashiers, otherwise a Manager override is needed.

1. Cashier enters item identifier for removal from sale.
2. System removes item and displays updated running total.
    - 2a. Item price exceeds void limit for Cashiers:
    - 1. System signals error, and suggests Manager Override.
    - 2. Cashier requests Manager Override, gets it, and repeats operation.

3-6b. Customer tells Cashier to cancel sale:

1. Cashier cancels sale on System.

3-6c. Cashier suspends the sale:

1. System records sale so that it is available for retrieval on any POS register.
2. System presents a "suspend receipt" that includes the line items, and a sale ID used to retrieve and resume the sale.

4a. the system supplied item price is not wanted (e.g., Customer complained about something and is offered a lower price):

1. Cashier requests approval from Manager.
2. Manager performs override operation.
3. Cashier enters manual override price.
4. System presents new price.

5a. System detects failure to communicate with external tax calculation system service:

1. System restarts the service on the POS node, and continues.
    - 1a. System detects that the service does not restart.
        1. System signals error.
        2. Cashier may manually calculate and enter the tax, or cancel the sale.

5b. Customer says they are eligible for a discount (e.g., employee, preferred customer):

1. Cashier signals discount request.
2. Cashier enters Customer identification.
3. System presents discount total, based on discount rules.

5c. Customer says they have credit in their account, to apply to the sale:

1. Cashier signals credit request.
2. Cashier enters Customer identification.
3. Systems applies credit up to price=0, and reduces remaining credit.

6a. Customer says they intended to pay by cash but don't have enough cash:

1. Cashier asks for alternate payment method.

1a. Customer tells Cashier to cancel sale. Cashier cancels sale on System.

7a. Paying by cash:

1. Cashier enters the cash amount tendered.
2. System presents the balance due, and releases the cash drawer.

3. Cashier deposits cash tendered and returns balance in cash to Customer.
4. System records the cash payment.

7b. Paying by credit:

1. Customer enters their credit account information.
2. System displays their payment for verification.
3. Cashier confirms.
    3a. Cashier cancels payment step:
    1. System reverts to "item entry" mode.
4. System sends payment authorization request to an external Payment Authorization Service System, and requests payment approval.
    4a. System detects failure to collaborate with external system:
    1. System signals error to Cashier.
    2. Cashier asks Customer for alternate payment.
5. System receives payment approval, signals approval to Cashier, and releases cash drawer (to insert signed credit payment receipt).
    5a. System receives payment denial:
    1. System signals denial to Cashier.
    2. Cashier asks Customer for alternate payment.
    5b. Timeout waiting for response.
    3. System signals timeout to Cashier.
    4. Cashier may try again, or ask Customer for alternate payment.
6. System records the credit payment, which includes the payment approval.
7. System presents credit payment signature input mechanism.
8. Cashier asks Customer for a credit payment signature. Customer enters signature.
9. If signature on paper receipt, Cashier places receipt in cash drawer and closes it.

7c. paying by check…

7d. Paying by debit…

7e. Cashier cancels payment step:

1. System reverts to "item entry" mode.

7f. Customer presents coupons:

1. Before handling payment, Cashier records each coupon and System reduces price as appropriate. System records the used coupons for accounting reasons.
    1a. Coupon entered is not for any purchased item:
    1. System signals error to Cashier.

9a. there are product rebates:

1. System presents the rebate forms and rebate receipts for each item with a rebate.

9b. Customer requests gift receipt (no prices visible):

1. Cashier requests gift receipt and System presents it.

9c. Printer out of paper.

1. If System can detect the fault, will signal the problem.
2. Cashier replaces paper.
3. Cashier requests another receipt.