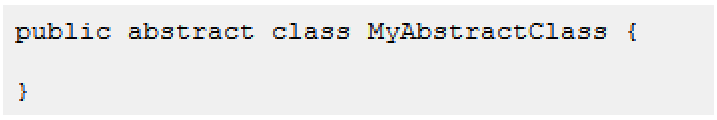
Task 1 [M1, 1.1]

Abstraction

Abstract Class in Java

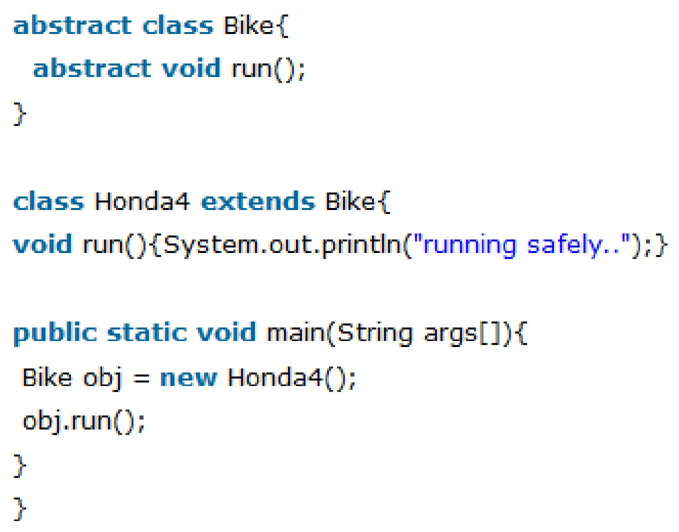
Abstract class in java are classes that are declared as abstract. Its methods are required to be implemented and is utilized using extended keyword. According to Jenkov (2015), programmer cannot instantiate an abstract class which means new instance of abstract class cannot be created. The key purpose of this type of class is to function as template of subclasses. Abstract class is declared using abstract keyword. Similarly, abstract methods are also declared using abstract keyword, these methods are abstract and does not have implementation. Example of declaration of abstract class and abstract method is provided below.



```java
public abstract class MyAbstractClass {

}
```

Figure 1Declaration of abstract class (source: http://tutorials.jenkov.com/)

In figure 1, abstract class is declared. As these classes cannot be instantiated, trying to create object of MyAbstractClass is invalid. Proper example of abstract class in java is provided here.



```java
abstract class Bike{
  abstract void run();
}

class Honda4 extends Bike{
void run(){System.out.println("running safely..");}

public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

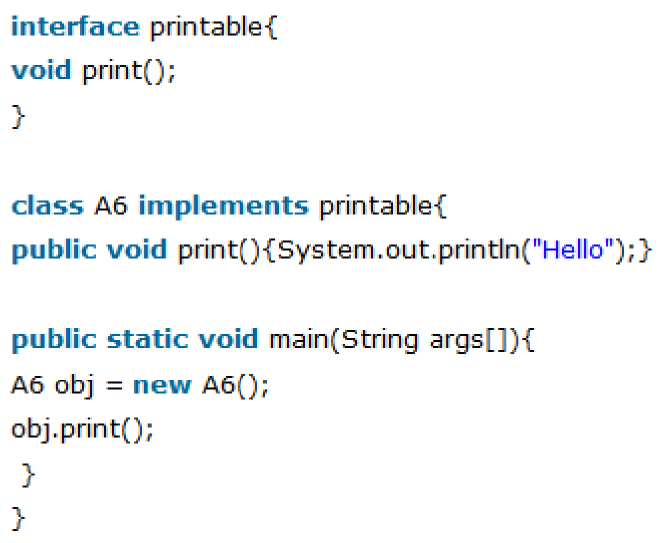
Figure 2 Implementation of abstract class in Java (Source: http://www.javatpoint.com/)

In given example of abstract class in Java (Figure 2) abstract class Bike is declared which has abstract method run. Honda4 class is declared which extends abstract class Bike hence it requires to implement each methods from class Bike. In main method, object of Honda4 is instantiated. Using object, run method is called. By this way, abstract allows to focus on what object does instead of how it does. Output from the above example is ‘running safely’.

Interface in Java

Java incorporates an idea called interfaces. A Java interface is somewhat like a class, with the exception of a Java interface can just contain method marks and fields. An interface can't contain a usage of the methods, just the mark (name, parameters and special cases) of the method. An interface in java is an outline of a class. It contains abstract methods and static constants.

JAVAPOINT (n.d.) writes, interface in java allows to achieve complete abstraction. Only abstract methods are allowed in interface unlike in abstract class where implementation of methods is possible. Interface also allows multiple inheritance in java.



```
interface printable{
void print();
}

class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
```

Figure 3Example of Interface (Source: http://www.javatpoint.com/)

In example above (figure 3), implementation of interface is illustrated. printable interface is declared using interface keyword and it contains method print () which is an abstract method. This interface printable is utilized by class A6 using keyword implements. Class A6 must implement exact same class print () from interface. Now in main method, instance of A6 is created and print method is called. Output from the above example is ‘Hello’.

Working Architecture of Java

To discuss working architecture of Java program, first how a Java program is created is discussed here. Programmer builds Java program by coding and implementing already available classes. Java Program in simplest form is just a group of classes. Developer follows specific syntax that Java compiler can understand. Once required code is written, Java file (with .java extension) is compiled. According to BFOIT (n.d.) the Java compiler turns Java file into a ‘.class’ file that a JVM (Java Virtual Machine) can interpret as shown in figure 4 below. This .class file is also described as byte code.

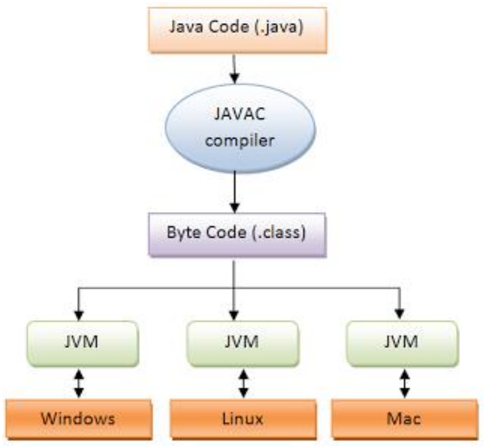
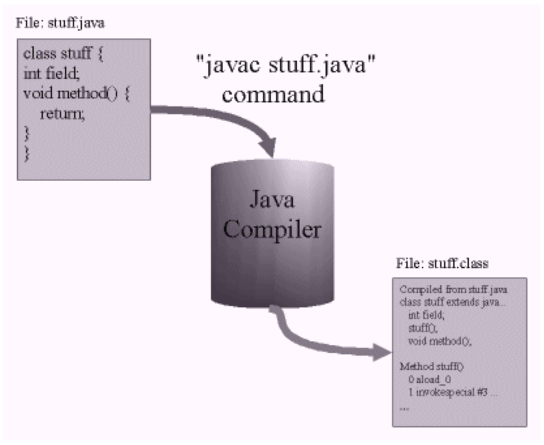


Figure 4 Basic working structure of Java Programming Language (Source: http://froilan-benito.blogspot.com)

Byte Code (.class)

Byte code (.class) is special instruction set that can be understood and executed by Java virtual machine. .class file is platform independent allowing, JVM to understand these files regardless of the platform (Benito, 2011). .class file (byte code) is generated compiler as shown in example below (figure 5). In given example (figure 5) stuff.java class is compiled by Java compiler and it is converted to stuff.class file. This stuff.class file will be understood by JVM.



```
class stuff {
int field;
void method() {
    return;
}
}
```

```
Compiled from stuff.java
class stuff extends java...
  int field;
  stuff();
  void method();

Method stuff()
  0 aload_0
  1 invokespecial #3 ...
...
```

Figure 5 Java Compiler (Source: http://www.bfoit.org/itp/JavaProgram.html)

JVM (Java Virtual Machine)

The Java virtual machine is a theoretical PC. Its detail characterizes certain components each Java virtual machine must have, however leaves numerous decisions to the planners of every execution. For instance, albeit all Java virtual machines must have the capacity to execute Java bytecodes, they might utilize any procedure to execute them. Additionally, the particular is sufficiently adaptable to permit a Java virtual machine to be executed either totally in software or to differing degrees in equipment. The adaptable way of the Java virtual machine's determination empowers it to be executed on a wide assortment of PCs and gadgets.

The JVM is utilized to execute Java applications. The Java compiler, javac, yields byte codes and places them into a .class record. The JVM then deciphers these byte codes, which can then be executed by any JVM usage, in this manner giving Java's cross-stage convey ability. JVM's main task is to load class files and understand/execute byte code (.class generated by compiler) as shown in figure 6.

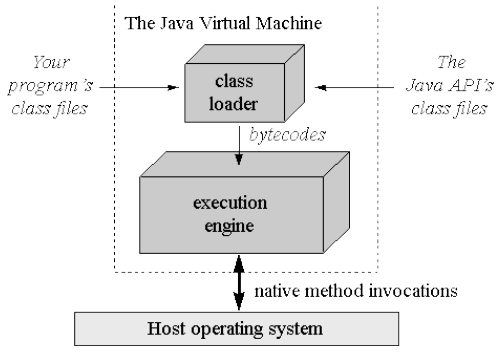


Figure 6Basic architecture of JVM (source: http://www.artima.com)

Execution engine part of JVM can vary in dissimilar implementations. Byte code is just interpreted one at a time in simplest form of execution engine. Whereas in faster (requires more memory) also known as JIT (Just in time) compiler, byte codes are compiled to native codes first. This native code is cached and reused when same method byte code is invoked. According to Venners (n.d.) another type of execution engine is adaptive optimizer. JVM interprets byte codes while also monitoring and optimizing heavily used area of code.

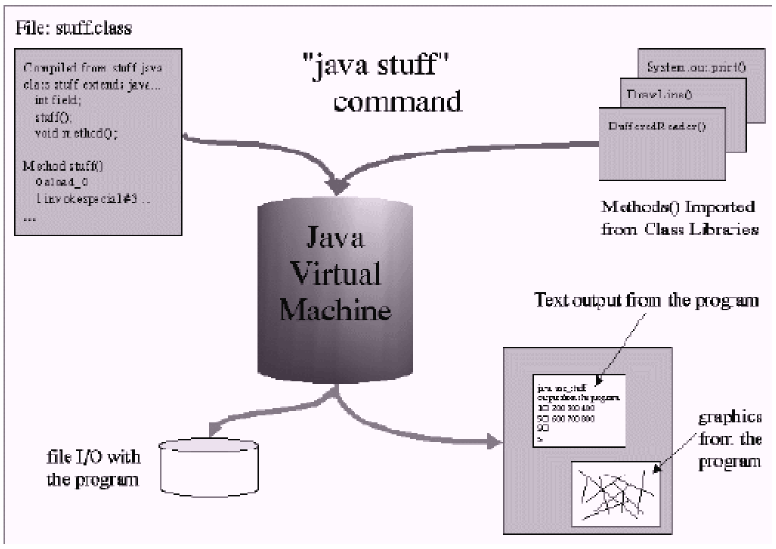

File: stuff.class
"java stuff" command
System.out.print()
Draw1.ine()
BufferedReader()
Methods() Imported from Class Libraries
Java Virtual Machine
Text output from the program
graphics from the program
file I/O with the program

Figure 7 Structure of Java Programming Language (Source: http://www.bfoit.org/itp/JavaProgram.html)

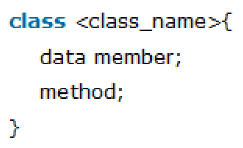
Figure 8 above demonstrates basis working structure of Java Virtual Machine (JVM). Stuff.class generated from Java compiler is understood by JVM and addition methods are imported from class libraries. JVM also interact with existing standard classes to support I/O with the program such as keyword, graphics stuffs, internet communication etc. Then finally, JVM provides Java application output as shown in figure 8.

Features of Java

Objects and Class

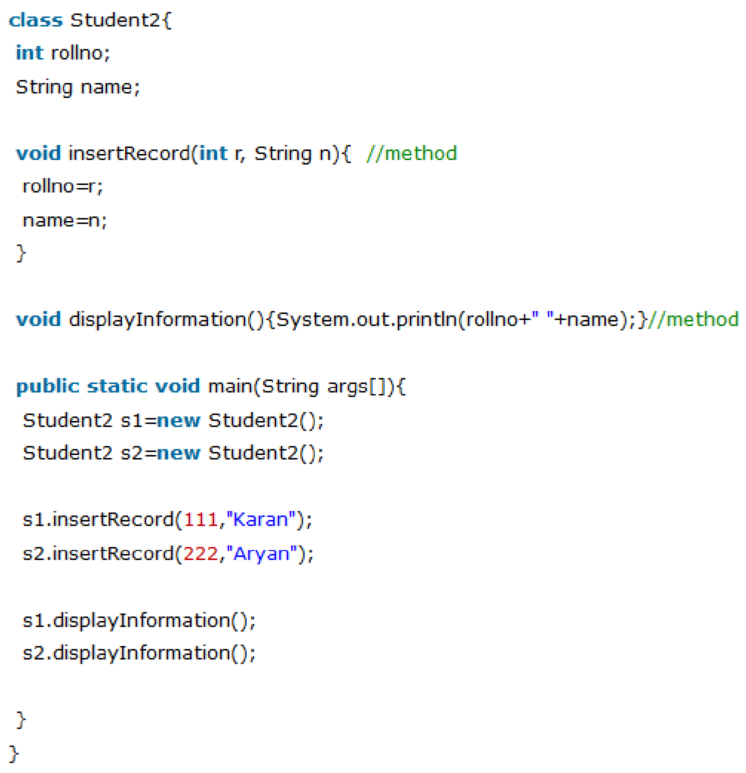
Object oriented nature of Java supports concepts like class, objects etc. An object is entities that demonstrates behavior and state. In real world scenario human, cars, dog etc. objects can be easily found. For example, a cat object describes its states- breed, color, and name and shows behavior like meowing, wagging, running etc. Similarly, in Java object have similar characteristics. In very simple words, JAVAPOINT (n.d.) has defined object as entity with state and behavior. Fields are utilized to store object states and methods to show behavior of the object. One of the key other characteristics of object in programming is identity. This identity is a unique id is used internally by Java virtual machine to identify each object.

Class is template from which objects are created. A class can contain methods, members, constructor, block, class and interface. Example of class and object is provided below.



```
class <class_name>{
    data member;
    method;
}
```

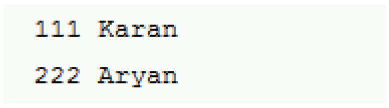
Figure 8Syntax for creating class



```
class Student2{
 int rollno;
 String name;

 void insertRecord(int r, String n){  //method
  rollno=r;
  name=n;
 }

 void displayInformation(){System.out.println(rollno+" "+name);}//method

 public static void main(String args[]){
  Student2 s1=new Student2();
  Student2 s2=new Student2();

  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");

  s1.displayInformation();
  s2.displayInformation();

 }
}
```

Figure 9Example of Class and Object (Source: http://www.javatpoint.com/object-and-class-in-java)

In the given example of class and objects in figure 9, class student is created using keyword class. It containts rollno and name properties and methods insertrecord () to insert record and displayInformation () to display record. ‘new’ keyword is used in main method to create object of class student 2. Using ‘object’ of class s1 and s2, values are passed to insertrecord() method using (.). In first object (s1) 111 and Karan is passed while in object 2 (s2) 222 and Aryan is passed. Now when displayInformation () is called, following output is achieved (figure 11).



```
111 Karan
222 Aryan
```

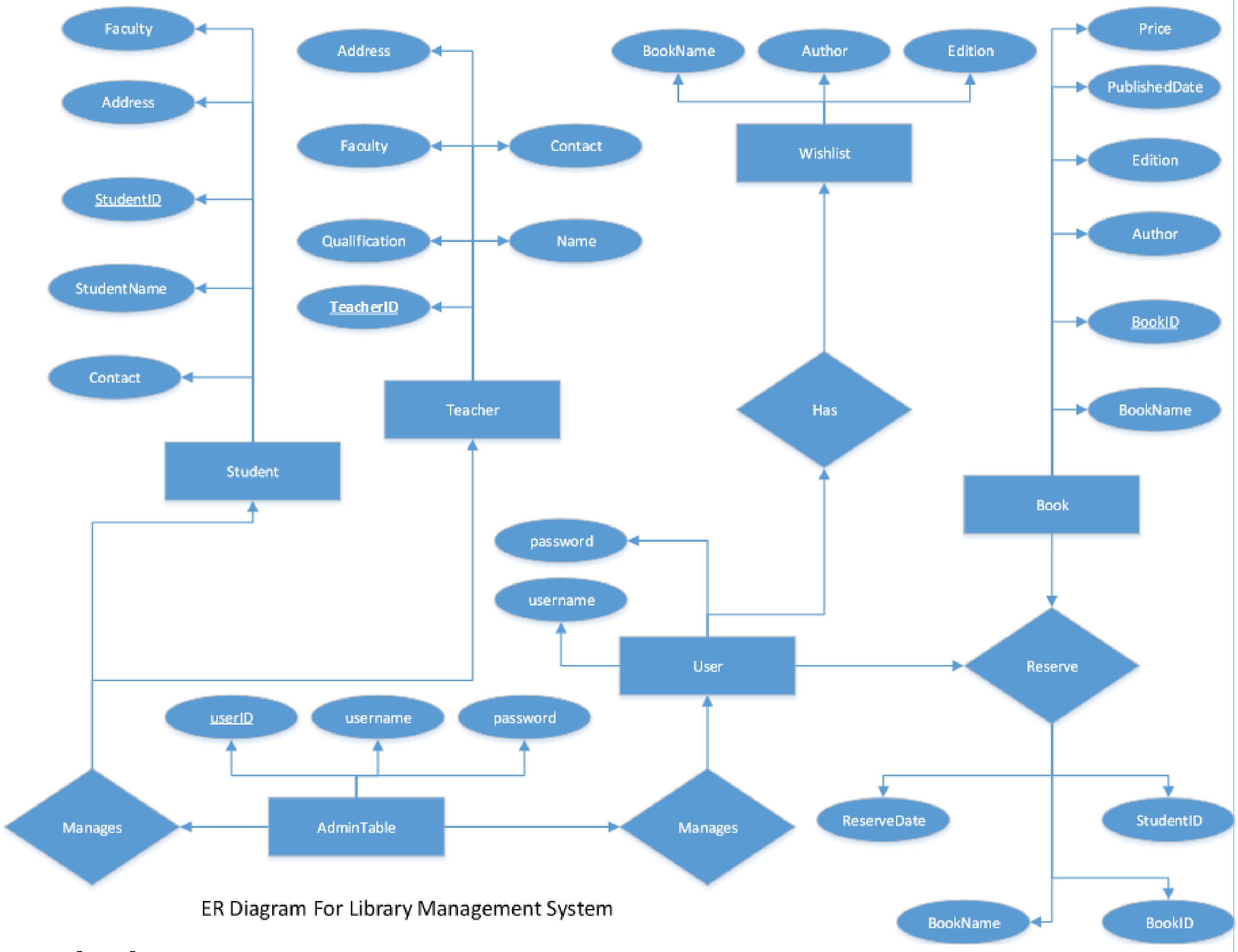
Figure 10Output from Example

<u>References</u>

- Benito, F.C. (2011) Java Programming Fundamentals I [Online] Available: http://froilan-benito.blogspot.com/2011/06/java-programming-fundamentals-i.html Accessed [2/6/2016]
- BFOIT (n.d.) A Java Program [Online] Available: http://www.bfoit.org/itp/JavaProgram.html Accessed [2/6/2016]
- JAVAPOINT (n.d.) Interface in Java [Online] Available: http://www.javatpoint.com/interface-in-java Accessed [2/5/2016]
- JAVAPOINT (n.d.) Object and Class in Java [Online] Available: http://www.javatpoint.com/object-and-class-in-java Accessed [2/6/2016]
- Jenkov, J. (2015) Java Abstract Classes [Online] Available: http://tutorials.jenkov.com/java/abstract-classes.html Accessed [2/5/2016]
- Venners, B. (n.d.) Introduction to Java's Architecture [Online] Available: http://www.artima.com/insidejvm/ed2/introarchP.html Accessed [2/6/2016]

Task 2 [M2, 2.1]

ER-Diagram



ER Diagram For Library Management System

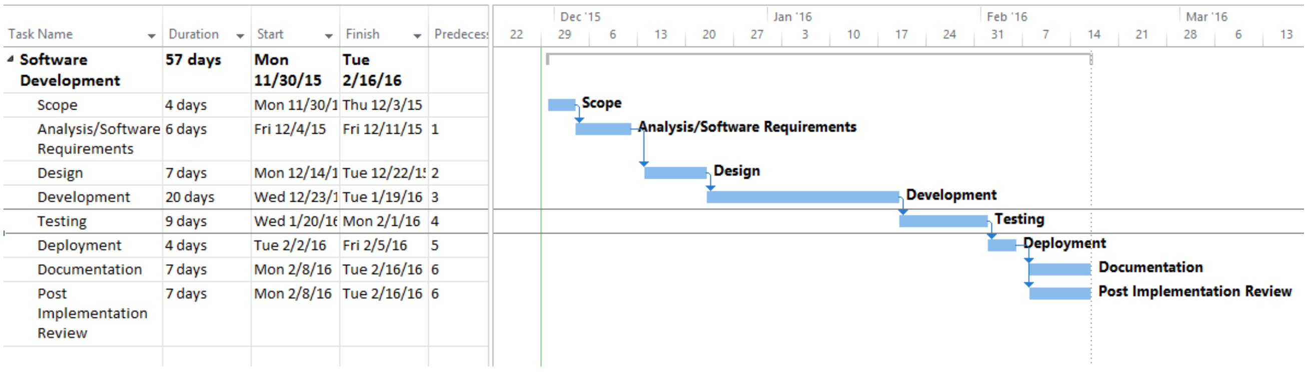
Flowchart

Flowchart for the system and description for each flowcharts are provided below.

Task 9 [M3, 4.5]

Task Breakdown Approach

Gantt chart



Task Name	Duration	Start	Finish	Predeces:
Software Development	**57 days**	**Mon 11/30/15**	**Tue 2/16/16**	
Scope	4 days	Mon 11/30/1	Thu 12/3/15	
Analysis/Software Requirements	6 days	Fri 12/4/15	Fri 12/11/15	1
Design	7 days	Mon 12/14/1	Tue 12/22/1:	2
Development	20 days	Wed 12/23/1	Tue 1/19/16	3
Testing	9 days	Wed 1/20/16	Mon 2/1/16	4
Deployment	4 days	Tue 2/2/16	Fri 2/5/16	5
Documentation	7 days	Mon 2/8/16	Tue 2/16/16	6
Post Implementation Review	7 days	Mon 2/8/16	Tue 2/16/16	6

Performed Activities

Task	Duration	Description
Scope Study	4 Days	During these activity, scope of the project is defined using various fact finding methods such as interview. Actual problem and viable solution is identified. This phase was completed in 4 days.
Software Requirement Analysis	6 Days	During this phase, all conceivable prerequisites are analyzed and documented. Software requirement Specification is created. All data collected during scope phase is analyzed to create SRS. This activity was completed in 6 days.
Design	7 Days	Based on the document created during scope study and SRS document from requirement analysis, how system can achieve business requirement is designed. Various tools and technologies. This activity took 7 days to complete and system design was completed using use case diagram, data flow diagram etc. UI of the system is designed as well.
Development	20 Days	After designing the system using various tools, implementation of design is started. This phase required 20 days to complete. Using various technology such as XML, Java etc. implementation was done based on the design created during design phase. Development Phase also contained unit testing along with implementation.

Testing	9 Days	Stress testing, compatibility testing and integration testing is performed in this phase after implementation is completed to ensure system is bug and error free. If any bug/error is found it is reported via documentation. Reported bug/error is handled before going to next phase of the software development.
Deployment	4 Days	After successful development and testing of developed system, it is deployed to library. Though it required only 1 day to install the system, 3 extra days were required to train library staffs.
Documentation	7 Days	During this phase, Technical and non-technical documentation is created. It took 7 days to complete this activity.
Post Implementation Review/Maintenance	7 Days	After successful completion of software planning, development and deployment as well as documentation, progress of system is monitored and user's complaints are noted. These complains are taken into consideration while developing updates/patches for the system.

Fact Finding Methods

To examine any program analyzer should do accumulate specifics and also almost all applicable facts. Precise facts might be gathered with aid of selected methods/ strategies. These kind of particular strategies to discovering facts in the program are usually termed as fact finding methods. To understand various issues faced by existing system in ISTM and understand actual problem and business requirements, Interview method is utilized.

This process is employed to gather the knowledge coming from organizations or perhaps men and women. Analyzer decides on the people who are related with the system with the meeting. Analyst utilizes face to face approach with people and collect their responses. The actual interviewer ought to prepare beforehand the type of queries he/ she is going to request and may get ready for you to reply any sort of dilemma. They should likewise go with a suitable place and time period which is cozy with the respondent. To collect required facts analyst interviewed ISMT library coordinator. Questions asked during interview and recorded response are provided below.

Asked Questions and Response

S.N.	**Question**	**Response**
1.	Who can be member of library?	Any active student can be member of library.

2.	Who is responsible for handling library?	Teacher/Library staff(s) is (are) responsible of handling library management.
3.	Is there any existing e-library system?	No, currently, library is managed in traditional way. Each record are kept in paper based database.
4.	Is there any back up system in existing library management system?	No.
5.	How many books a member can borrow?	Only two books can be borrowed at a time. Members are required to return they have taken before borrowing new books.
6.	What problems library is facing from existing system?	Available, borrowed books are required to be searched manually from database, fines are required to be calculated manually. Anyone can access the library database.
7.	Does member have to pay for borrowing books?	No.
8.	Does member have to pay fine for exceeding allowed time? How much?	Yes. Members will be charged Rs.10 for each exceeded days.
9.	For how many days members can keep borrowed books without paying fines?	Two weeks (fourteen days)
10.	Does library management wants to allow member to add wish lists?	Yes.
11.	What library management expects from new system?	Administrator should be able to add/manage/delete books/members/users. System should allow to filter unavailable books and administrator should be able to calculate fines automatically. General members should able to add wish list and administrators should be able to view them.

Suggestions

To understand the issues in existing system and understand the business requirement of the system, Interview fact finding method is utilized. Interview response shows that existing system has failed to satisfy requirement for the library management. It is suggested to develop mobile based library management system due to its portability and availability. As most of the students/staffs carry android based smart phones, developing android application is a feasible solution. This also allows to take advantages of flexibility while programming with Java programming language. Fact findings shows following modules are required in new system, Authentication and authorization module, Books module, user management module, reserve books module, fine module, wish list module etc.