CHAPTER 3

PROJECT ANALYSIS AND DESIGN

3.1 Project Timeline & task distribution

Gantt chart:

A **Gantt chart** is a type of bar chart, developed by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project.

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

- What the various activities are.
- When each activity begins and ends.
- How long each activity is scheduled to last.
- Where activities overlap with other activities, and by how much.
- The start and end date of the whole project.

.

Table 3.1 Task Distribution

Tasks List	**Assigned to**	**State**
Defining Project goal	Purvisha Khunt Dhruv Tank Jignesh Lad	Completed
Plan project	Purvisha Khunt Dhruv Tank Jignesh Lad	Completed
Analysis	Purvisha Khunt Dhruv Tank Jignesh Lad	Completed
Gui design	Purvisha Khunt Jignesh Lad	Completed
Implementation	Purvisha Khunt Dhruv Tank	Completed
Testing	Dhruv Tank Jignesh Lad	Completed
Final reports and presentation	Purvisha Khunt Dhruv Tank Jignesh Lad	Completed

3.2 Development Methodology

This section describes the project as per the various stages of the Software Development life cycle. The model of software development life cycle used in this project is the waterfall method. The Waterfall Method is comprised of a series of very definite phases, each one run intended to be started sequentially only after the last has been completed, with one or more tangible deliverables produced at the end of each phase of the waterfall method of SDLC. Essentially, it starts with a heavy, documented, requirements planning phase that outlines all the requirements for the project, followed by sequential phases of design, coding, test-casing, optional documentation, verification (alpha-testing), validation (beta-testing), and finally deployment/release.

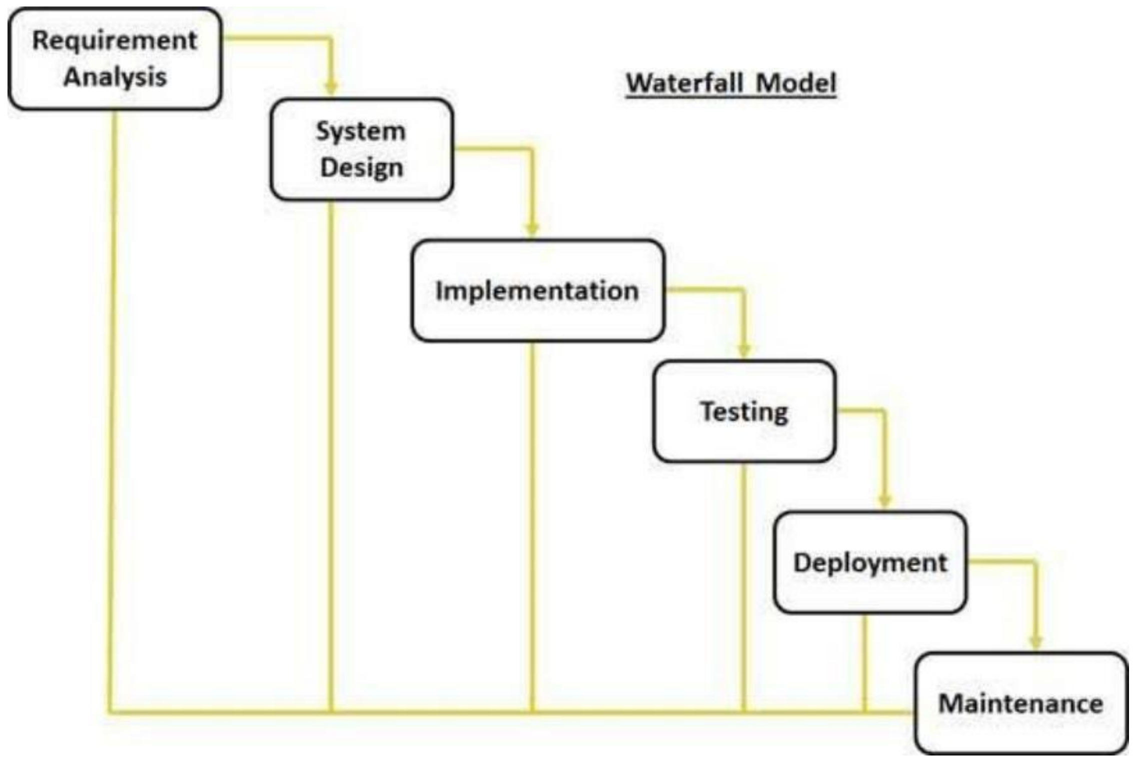

Figure 3.2: Waterfall Model

Requirement Analysis: Existing system is time consuming and requires more amount of cost. The confidentiality of data provided by existing system is also poor. Thus all the above stated problems are resolved in the proposed system. It will provide better data confidentiality and is less time consuming.

System Design: It includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language. The output of the design phase is a design document that acts as an input for all the subsequent SDLC phases. The design of this software is simple and user friendly containing main activities, namely: (a) Interactive User Interface (b) About Form (c) Class Diagram Editor (d) Activity Diagram Editor.

Coding/Implementation: The project is implemented using the C# language in Visual Studio. Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

Testing: It includes detection of errors in the application. The testing process starts with a test plan that recognizes test-related activities, such as test case generation, testing criteria, and resource allocation for testing. The code is tested and mapped against the design document created in the design phase. The output of the testing phase is a test report containing errors that occurred while testing the application. Testing has been done for each of the individual activities of the project.

Maintenance: It includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed at the customer location. The maintenance phase also includes handling the residual errors that may exist in the software even after the testing phase [4].

3.3 Design Details

1. Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

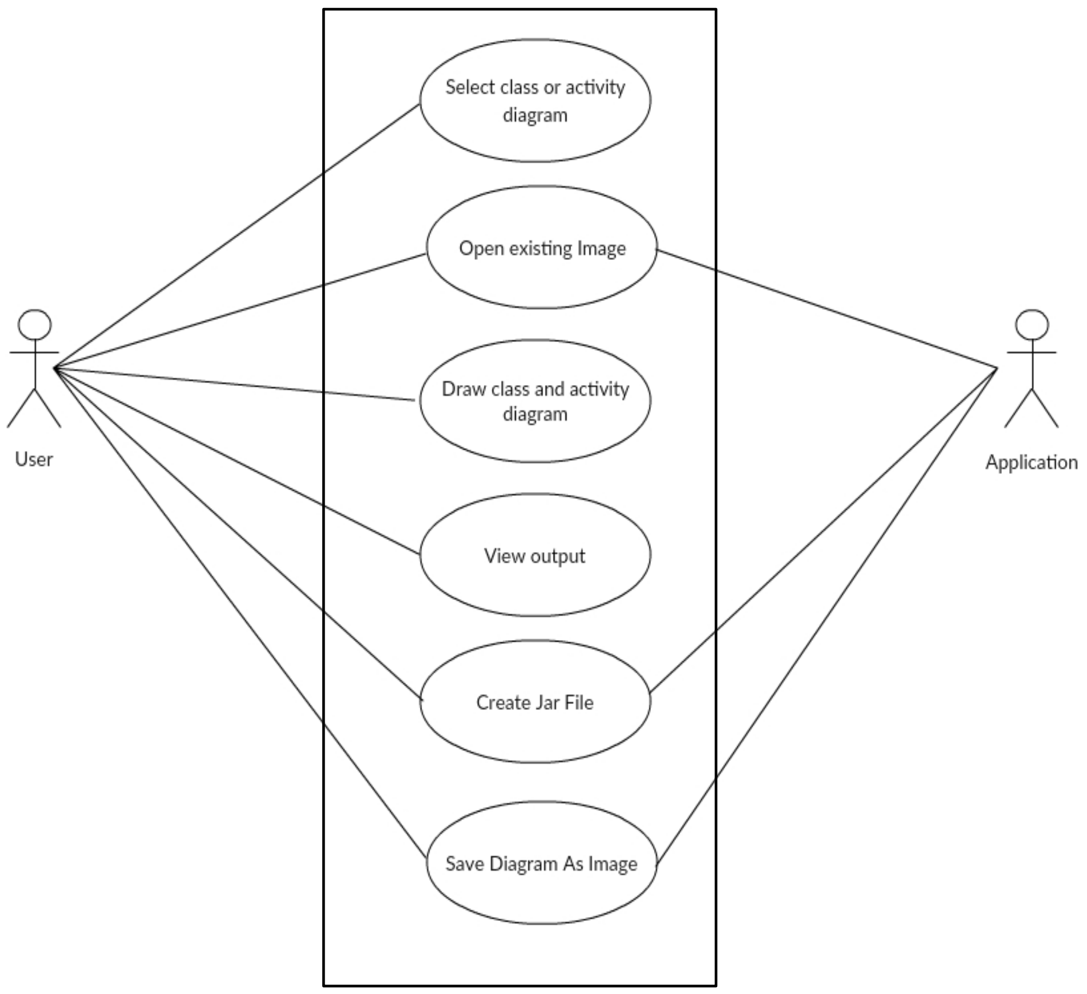

Figure 3.2: UseCase Diagram

2. Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

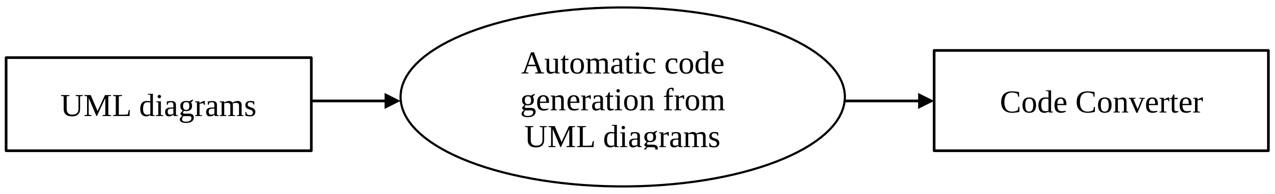

Figure 3.4: Data flow level 0

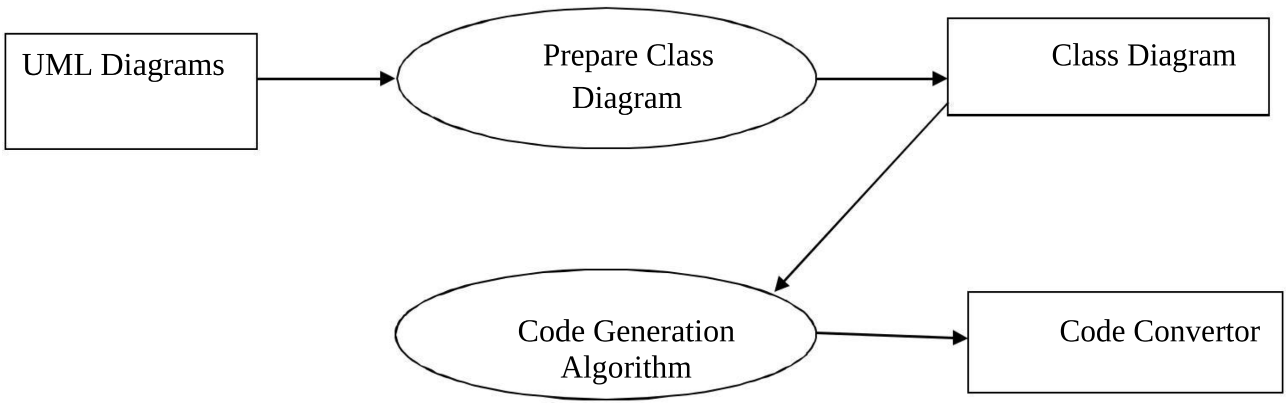

Figure 3.5: Data flow level 1

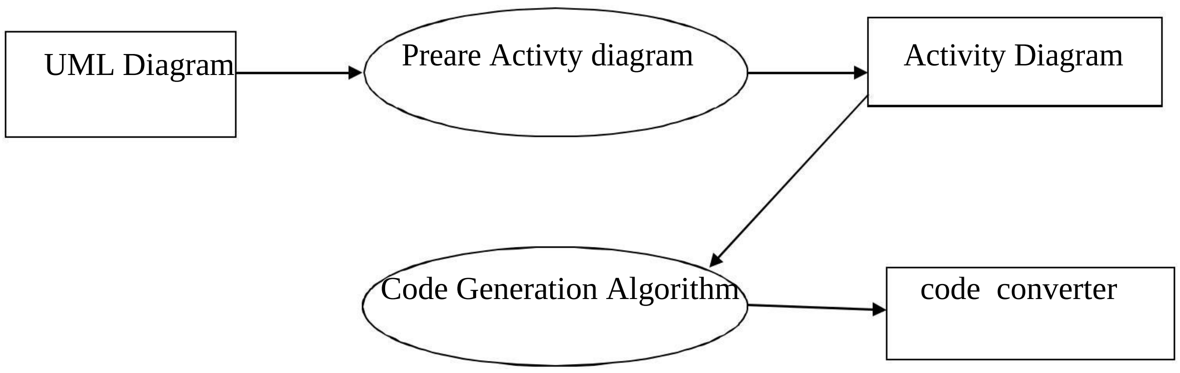

Figure 3.6: Data flow level 2

3. Control Flow Diagram

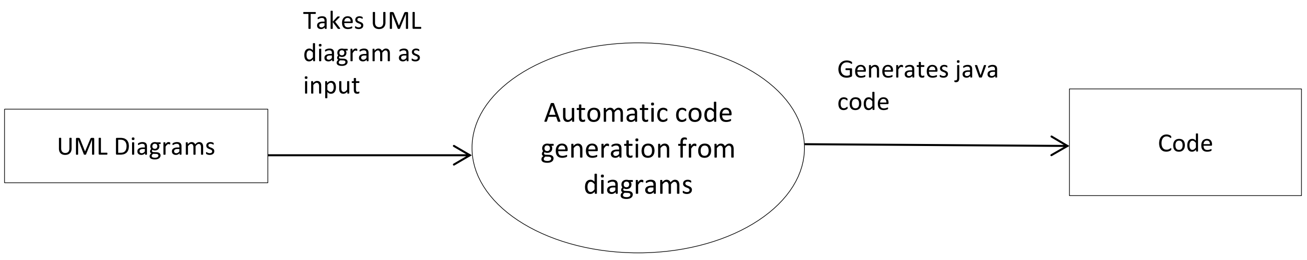
Takes UML diagram as input
UML Diagrams
Automatic code generation from diagrams
Generates java code
Code

Figure 3.5: Control flow level 0

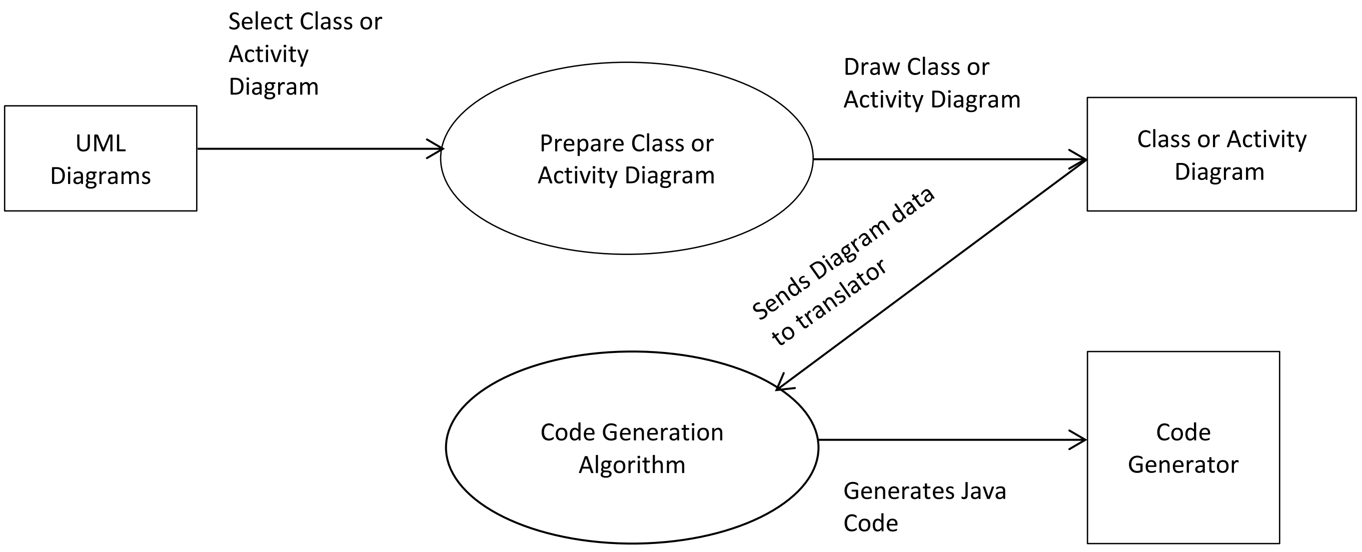
Select Class or Activity Diagram
UML Diagrams
Prepare Class or Activity Diagram
Draw Class or Activity Diagram
Class or Activity Diagram
Sends Diagram data to translator
Code Generation Algorithm
Generates Java Code
Code Generator

Figure 3.6: Control flow level 1