**Acknowledgement**

We are pleased to present **“Heart Disease Prediction”** projectand takethis opportunity to express our profound gratitude to all those people who helped us in completion of this project.

We thank our college for providing us with excellent facilities that helped us to complete and present this project. We would also like to thank the staff members and lab assistants for permitting us to use computers in the lab as and when required.

We express our deepest gratitude towards our project guide for her valuable and timely advice during the various phases in our project. We would also like to thank her for providing us with all proper facilities and support as the project co-coordinator. We would like to thank her for support, patience and faith in our capabilities and for giving us flexibility in terms of working and reporting schedules.

We would like to thank all our friends for their smiles and friendship making the college life enjoyable and memorable and family members who always stood beside us and provided the utmost important moral support. Finally we would like to thank everyone who has helped us directly or indirectly in our project.

**PROJECT OVERVIEW**

Introduction

A major challenge facing healthcare organizations (hospitals, medical centers) is the provision of quality services at affordable costs. Quality service implies diagnosing patients correctly and administering treatments that are effective. Poor clinical decisions can lead to disastrous consequences which are therefore unacceptable. Hospitals must also minimize the cost of clinical tests. They can achieve these results by employing appropriate computer-based information and/or decision support systems. Most hospitals today employ some sort of hospital information systems to manage their healthcare or patient data. These systems typically generate huge amounts of data which take the form of numbers, text, charts and images. Unfortunately, these data are rarely used to support clinical decision making. There is a wealth of hidden information in these data that is largely untapped. This raises an important question: “How can we turn data into useful information that can enable healthcare practitioners to make intelligent clinical decisions?” Although data mining has been around for more than two decades, its potential is only being realized now. Data mining combines statistical analysis, machine learning and database technology to extract hidden patterns and relationships from large databases. The two most common modeling objectives are classification and prediction. Classification models predict categorical labels (discrete, unordered) while prediction models predict continuous-valued functions. Decision Trees and Neural Networks use classification algorithms while Regression, Association Rules and Clustering use prediction algorithms.

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning). Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is:

**A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.**

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don’t constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So, the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

Modules and their Description

The system comprises of 2 major modules as follows:

**Admin Module**

1. Add Training Data
2. Add Doctor Details
3. View User Details
4. View Feedback
5. View Doc Details
6. View Training Data

**User Module**

1. Register (With Details like Age, Sex, etc.)
2. Check Heart (By providing Details like

* Age in Year
* Gender
* Chest Pain Type
* Fasting Blood Sugar
* Resting Electrographic Results(Restecg)
* Exercise Induced Angina(Exang)
* The slope of the peak exercise ST segment
* CA – Number of major vessels colored by fluoroscopy
* Thal
* Trest Blood Pressure
* Serum Cholesterol
* Maximum heart rate achieved(Thalach)
* ST depression induced by exercise(Oldpeak)

1. System will accordingly view Doctor to consult.
2. Give Feedback
3. View Doctor

Existing System & Proposed System

* **Problem with current scenario**

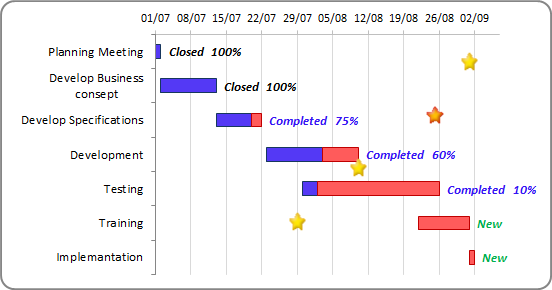
Many hospital information systems are designed to support patient billing, inventory management and generation of simple statistics. Some hospitals use decision support systems, but they are largely limited. They can answer simple queries like “What is the average age of patients who have heart disease?”, “How many surgeries had resulted in hospital stays longer than 10 days?”, “Identify the female patients who are single, above 30 years old, and who have been treated for cancer.” However, they cannot answer complex queries like “Identify the important Preoperative predictors that increase the length of hospital stay”, “Given patient records on cancer, should treatment include chemotherapy alone, radiation alone, or both chemotherapy and radiation?”, and “Given patient records, predict the probability of patients getting a heart disease.”

* **Drawbacks of the existing system**
* Maintenance of the system is very difficult.
* There is a possibility for getting inaccurate results.
* User friendliness is very less.
* It consumes more time for processing the activities.

**PROPOSED SYSTEM**

* Considering the anomalies in the existing system computerization of the whole activity is being suggested after initial analysis.
* It might have happened so many times that you or someone yours need doctors help immediately, but they are not available due to some reason.
* The Heart Disease Prediction application is an end user support and online consultation project.
* Here, we propose an android application that allows users to get instant guidance on their heart disease through an intelligent system online.
* The application is fed with various details and the heart disease associated with those details.
* The application allows user to share their heart related issues.
* It then processes user specific details to check for various illness that could be associated with it.
* Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with patient’s details.
* We use Random Forest Classifier to predict the illness with the help of multiple health parameters.
* Based on result, system automatically shows the result specific doctor’s for further treatment.
* The system allows user to view doctor’s details.
* The system can be use in case of emergency.

Gantt Chart



**PROJECT DESIGN**

E-R Diagram

User Registration

provides

**User Login**

**Admin Login**

access

**Heart Disease Prediction**

access

Use Case Diagram

User/Patient

Admin

Sequence Diagram

🚺

Heart Disease Prediction

Main Page

User

Enters User\_id & Password

Validates User

Invalid Login

**Screenshot (27)** - - - - - - - - - - - - - - - - - - - - - - - - - - - -

New Registration

If Valid, then Displays Main page

Specifies the Details to check Heart Disease

Ask multiple questions based on Disease

Checks the accurate result

Searches/Views Doctor accordingly

Communicates with the doctor

Feedback

🚺

Main Page

Heart Disease Prediction

Admin

Enters Admin\_id & Password

Validates Admin

Invalid Login

**Screenshot (27)** - - - - - - - - - - - - - - - - - - - - - - - - - - - -

If Valid, then Displays Main page

Adds Training Data

Training Data Added

Adds Doctor/Disease Details

Details Added Successfully

Views Doctor/Disease/User Details

Retrieves the Respective Details

View Feedback

Activity Diagram

User

START 





Invalid Login

If Valid

 STOP

User

Login

Access to Main Page

Specifies the Details to check Heart

Checks the accurate result

View Doctor

Feedback

Admin

START 





Invalid Login

If Valid

 STOP

Admin

Login

Access to Main Page

Adds Training Data

Adds Doctor/Disease Details

Views Doctor/Disease/Patient Details

Manages Security Issues

Feedback

Class Diagram

ADMIN

- Admin\_id : String

- Password : String

+ Login()

+ btn\_Click ()

+ Logout ()

USER

- User\_id : String

- Password : String

+ Login()

+ btn\_Click ()

+ Logout ()

New Registration

- Fname : String

- Lname : String

- DOB : Int

- Gender : String

- Address : String

- Mobile No. : Int

- Email id : String

- User id : String

- Password : String

Data Flow Diagram

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD’s is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data

Data flow

Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD’s:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

#### SAILENT FEATURES OF DFD’s

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the data flows take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transform them regardless of actual physical form.

**NEW LOGICAL**:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system.

**RULES GOVERNING THE DFD’S**

PROCESS

1. No process can have only outputs.
2. No process can have only inputs. If an object has only inputs than it must be a sink.
3. A process has a verb phrase label.

**DATA STORE**

1. Data cannot move directly from one data store to another data store, a process must move data.
2. Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3. A data store has a noun phrase label.

**SOURCE OR SINK**

The origin and /or destination of data.

1. Data cannot move direly from a source to sink it must be moved by a process
2. A source and /or sink has a noun phrase land

DATA FLOW

1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later it usually indicated however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4. A Data flow to a data store means update (delete or change).
5. A data Flow from a data store means retrieve or use.

Data Flow Diagrams

Heart Disease Prediction DB

0.0

User

Database

DATABASE DETAIL

Query

Process

Request

1.0

User

Query

Database

Feedback For

User

Check for user

Requirement

User need

Relevant

Data

1.1

LEVEL 1 DFD

Accept

Query

2.0

User

Check Availability of or for query processing

Process

Query

Give request to user

Via Heart Disease Prediction DB

Give info about DB

2.1

2.2

Query

LEVEL 2 DFD: PREDICTION

Snapshots

**PROJECT IMPLEMENTATION**

Project Implementation Technology

The Project application is loaded in Android Studio. We used Android Studio for Design and coding of project. Created and maintained all databases into SQL Server, in that we create tables, write query for store data or record of project.

* **Hardware Requirement:**

1. **Laptop or PC**

* i3 Processor Based Computer or higher
* 1GB RAM
* 5 GB Hard Disk

1. **Android Phone or Tablet**

* 1.2 Quad core Processor or higher
* 1 GB RAM
* **Software Requirement:**

1. **Laptop or PC**

* Windows 7 or higher.
* Java
* Android Studio

1. **Android Phone or Tablet**

* Android v5.0 or Higher

# **Introduction to Android**

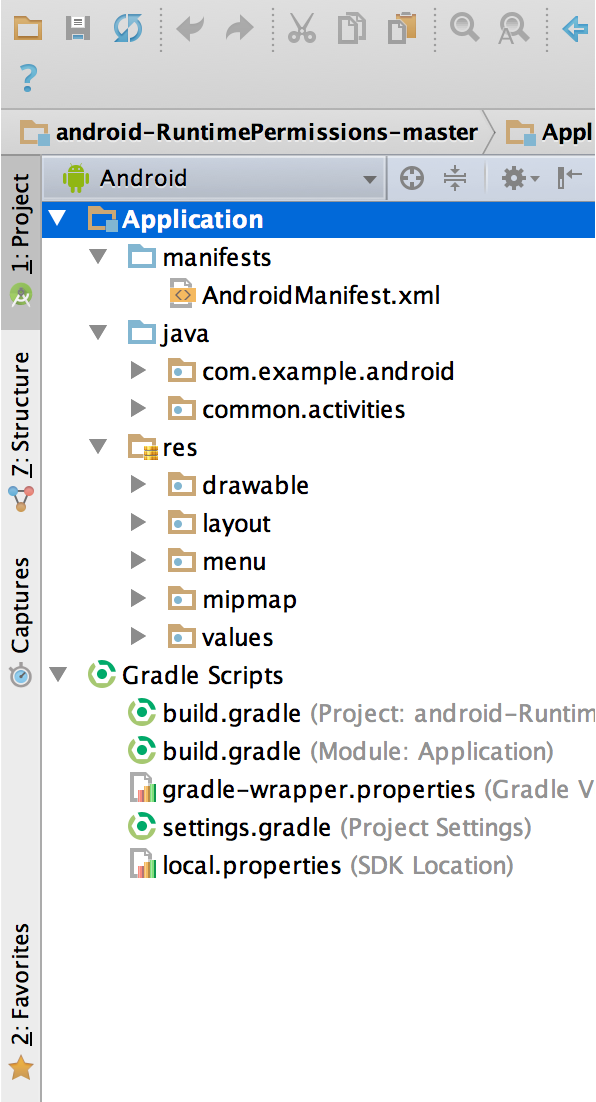
Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on [IntelliJ IDEA](https://www.jetbrains.com/idea/). On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

* A flexible Gradle-based build system
* A fast and feature-rich emulator
* A unified environment where you can develop for all Android devices
* Instant Run to push changes to your running app without building a new APK
* Code templates and GitHub integration to help you build common app features and import sample code
* Extensive testing tools and frameworks
* Lint tools to catch performance, usability, version compatibility, and other problems
* C++ and NDK support
* Built-in support for [Google Cloud Platform](http://developers.google.com/cloud/devtools/android_studio_templates/), making it easy to integrate Google Cloud Messaging and App Engine.

**Project Structure**

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

* Android app modules
* Library modules
* Google App Engine modules

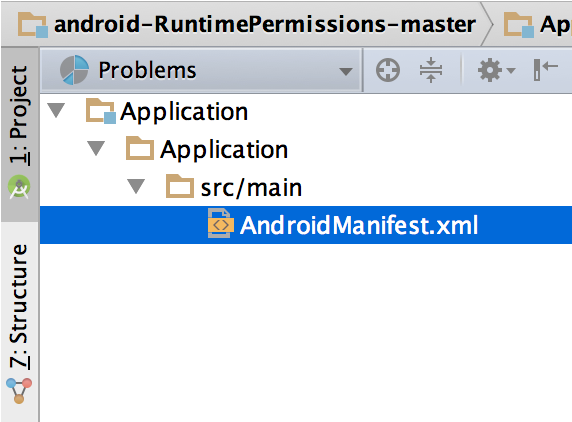


By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

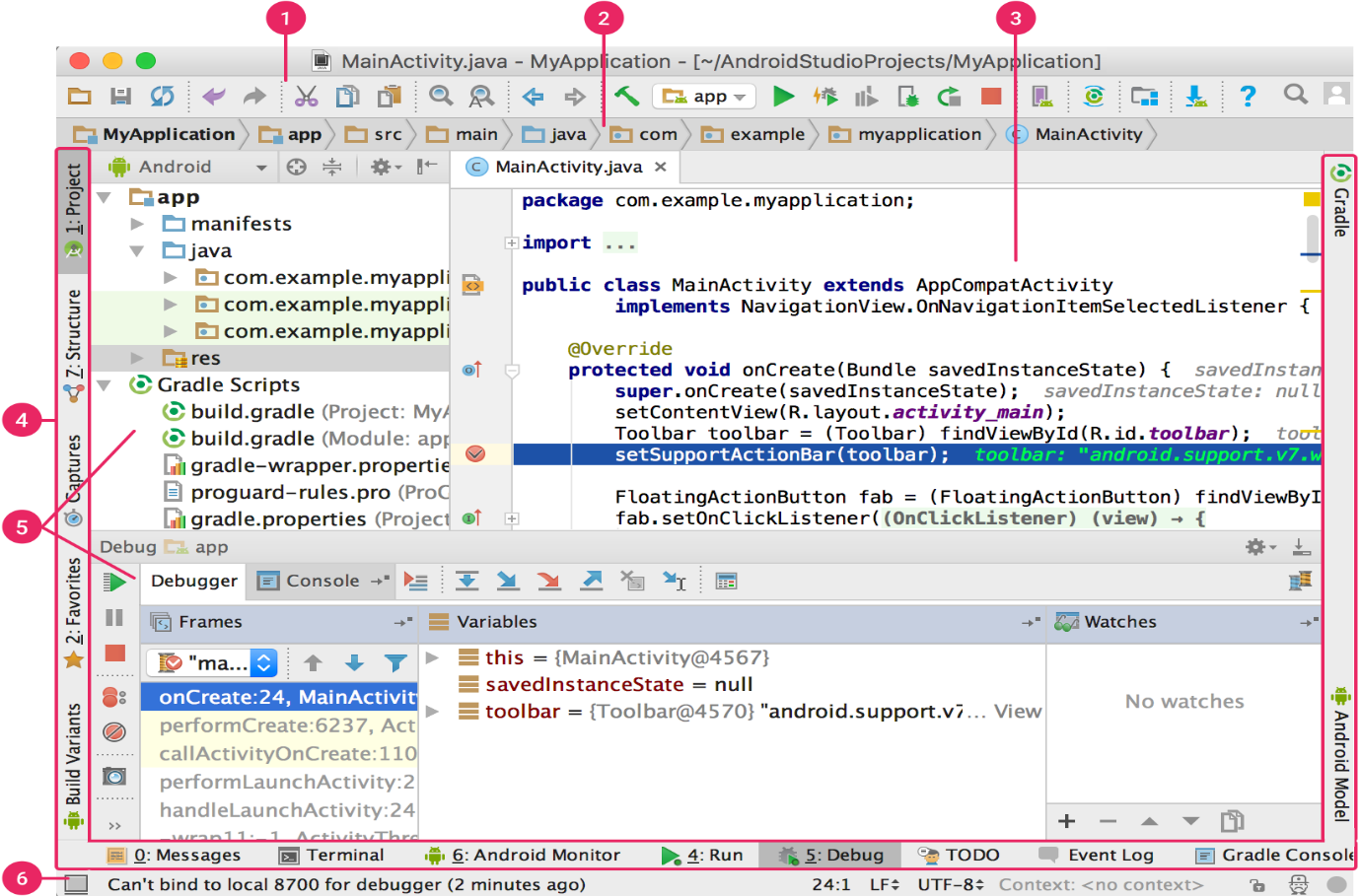
* **manifests**: Contains the AndroidManifest.xml file.
* **java**: Contains the Java source code files, including JUnit test code.
* **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as**Android**).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

## **The User Interface**

1. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.



You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

### **Tool Windows**

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

* To expand or collapse a tool window, click the tool’s name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.
* To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.
* To show or hide the entire tool window bar, click the window icon https://developer.android.com/studio/images/intro/window-icon_2-1_2x.png in the bottom left-hand corner of the Android Studio window.
* To locate a specific tool window, hover over the window icon and select the tool window from the menu.

### **Navigation**

Here are some tips to help you move around Android Studio.

* Switch between your recently accessed files using the *Recent Files* action. Press **Control+E** (**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
* View the structure of the current file using the *File Structure* action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
* Search for and navigate to a specific class in your project using the *Navigate to Class* action. Bring up the action by pressing **Control+N**(**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.
* Navigate to a file or folder using the *Navigate to File* action. Bring up the Navigate to File action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a / at the end of your expression.
* Navigate to a method or field by name using the *Navigate to Symbol* action. Bring up the Navigate to Symbol action by pressing **Control+Shift+Alt+N**(**Command+Shift+Alt+O** on a Mac).
* Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7**

## **Gradle Build System**

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the [Android plugin for Gradle](https://developer.android.com/tools/revisions/gradle-plugin.html). This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

* Customize, configure, and extend the build process.
* Create multiple APKs for your app, with different features using the same project and modules.
* Reuse code and resources across sourcesets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are namedbuild.gradle. They are plain text files that use [Groovy](http://groovy.codehaus.org/) syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

### **Multiple APK Support**

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the hdpi and mdpi screen densities, while still considering them a single variant and allowing them to share test APK, javac, dx, and ProGuard settings.

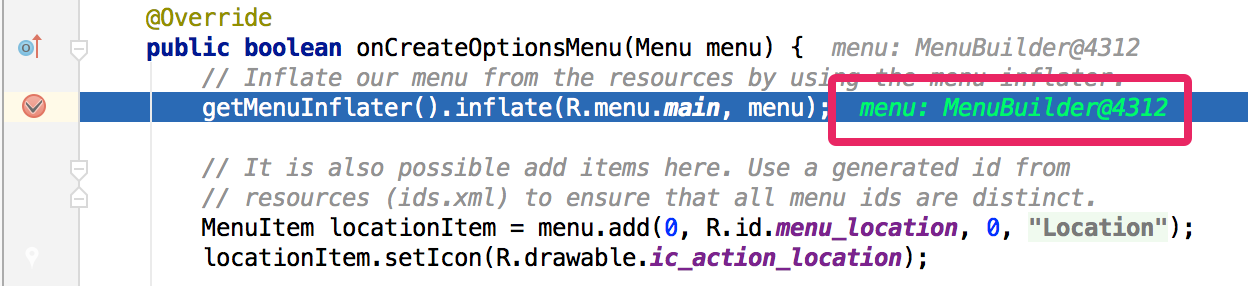
## **Debug and Profile Tools**

Android Studio assists you in debugging and improving the performance of your code, including inline debugging and performance analysis tools.

### **Inline debugging**

Use inline debugging to enhance your code walk-throughs in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

* Inline variable values
* Referring objects that reference a selected object
* Method return values
* Lambda and operator expressions
* Tooltip values



### **Performance monitors**

Android Studio provides performance monitors so you can more easily track your app’s memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests. With your app running on a device or emulator, open the **Android Monitor** tool window, and then click the **Monitors** tab.

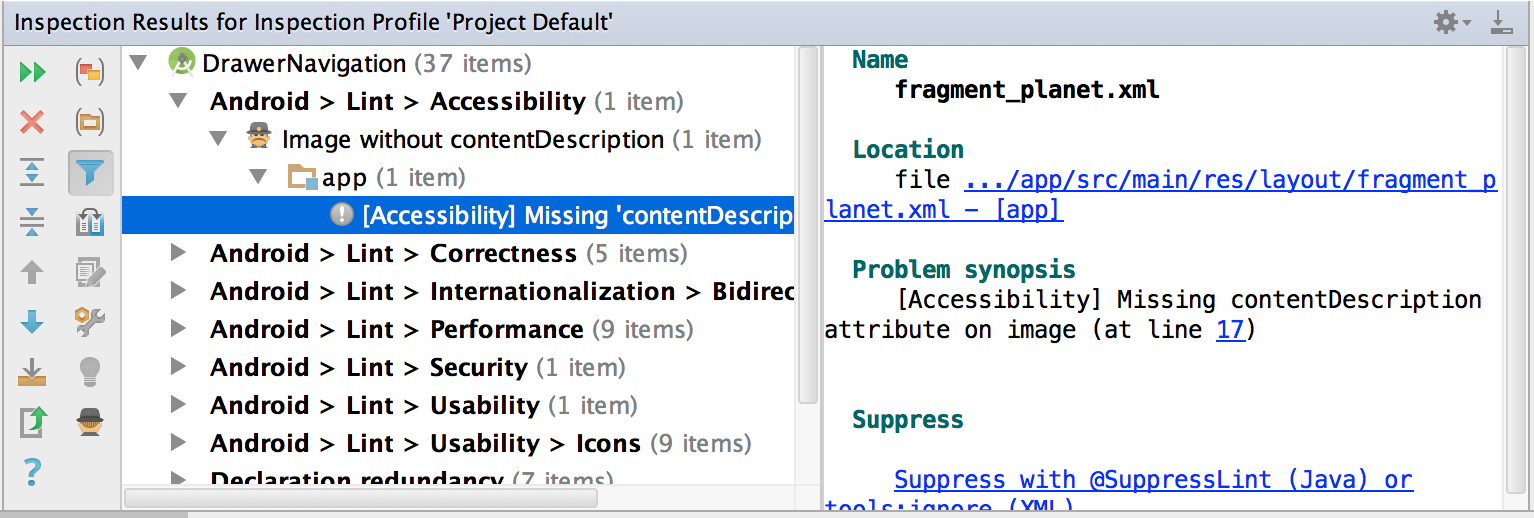
### **Allocation tracker**

Android Studio allows you to track memory allocation as it monitors memory use. Tracking memory allocation allows you to monitor where objects are being allocated when you perform certain actions. Knowing these allocations enables you to optimize your app’s performance and memory use by adjusting the method calls related to those actions.

### **Code inspections**

Whenever you compile your program, Android Studio automatically runs configured [Lint](https://developer.android.com/tools/help/lint.html) and other [IDE inspections](https://www.jetbrains.com/help/idea/2016.1/code-inspections.html?origin=old_help&search=inspection) to help you easily identify and correct problems with the structural quality of your code.

The Lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.



CODING

FEASIBILITY REPORT

Feasibility Studyis a high level capsule version of the entire process intended to answer a number of questions like: What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? Feasibility study is conducted once the problem clearly understood. Feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system.

The following feasibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

* Technical Feasibility
* Economic Feasibility
* Operational Feasibility

In this phase, we study the feasibility of all proposed systems, and pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows.

* Technical Feasibility

In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not.

Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds:

* All necessary technology exists to develop the system.
* This system is too flexible and it can be expanded further.
* This system can give guarantees of accuracy, ease of use, reliability and the data security.
* This system can give instant response to inquire.

Our project is technically feasible because, all the technology needed for our project is readily available.

**Operating System :** Android v5.0

or Higher (For Android Devices)

**Languages :** JAVA

**Database System :** MS-SQL Server

**Documentation Tool :** MS - Word

* Economic Feasibility

Economically, this project is completely feasible because it requires no extra financial investment and with respect to time, it’s completely possible to complete this project in 6 months.

In this step, we verify which proposal is more economical. We compare the financial benefits of the new system with the investment. The new system is economically feasible only when the financial benefits are more than the investments and expenditure. Economic Feasibility determines whether the project goal can be within the resource limits allocated to it or not. It must determine whether it is worthwhile to process with the entire project or whether the benefits obtained from the new system are not worth the costs. Financial benefits must be equal or exceed the costs. In this issue, we should consider:

* The cost to conduct a full system investigation.
* The cost of h/w and s/w for the class of application being considered.
* The development tool.
* The cost of maintenance etc...

Our project is economically feasible because the cost of development is very minimal when compared to financial benefits of the application.

* Operational Feasibility

In this step, we verify different operational factors of the proposed systems like man-power, time etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibilitydetermines if the proposed system satisfied user objectives could be fitted into the current system operation.

* The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
* The clients have been involved in the planning and development of the system.
* The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the time requirements and personnel requirements are satisfied. We are a team of four members and we worked on this project for three working months.

**TESTING**

As the project is on bit large scale, we always need testing to make it successful. If each components work properly in all respect and gives desired output for all kind of inputs then project is said to be successful. So the conclusion is-to make the project successful, it needs to be tested.

The testing done here was System Testing checking whether the user requirements were satisfied. The code for the new system has been written completely using JAVA as the coding language and Android Studio as the interface for front-end designing. The new system has been tested well with the help of the users and all the applications have been verified from every nook and corner of the user.

Although some applications were found to be erroneous these applications have been corrected before being implemented. The flow of the forms has been found to be very much in accordance with the actual flow of data.

Levels of Testing

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are:

Client Needs Acceptance Testing

Requirements System Testing

Design Integration Testing

Code Unit Testing

A series of testing is done for the proposed system before the system is ready for the user acceptance testing.

The steps involved in Testing are:

* **Unit Testing**

Unit testing focuses verification efforts on the smallest unit of the software design**,** the module**.** This is also known as “Module Testing”**.** The modules are tested separately**.** This testing carried out during programming stage itself**.** In this testing each module is found to be working satisfactorily as regards to the expected output from the module**.**

* **Integration Testing**

Data can be grossed across an interface**;** one module can have adverse efforts on another**.** Integration testing is systematic testing for construction the program structure while at the same time conducting tests to uncover errors associated with in the interface. The objective is to take unit tested modules and build a program structure**.** All the modules are combined and tested as a whole**.** Here correction is difficult because the isolation of cause is complicate by the vast expense of the entire program. Thus in the integration testing stop**,** all the errors uncovered are corrected for the text testing steps**.**

* **System testing**

System testing is the stage of implementation that is aimed at ensuring that the system works accurately and efficiently for live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, then goal will be successfully achieved.

* **Validation Testing**

At the conclusion of integration testing software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins**,** validation test begins**.** Validation test can be defined in many ways**.** But the simple definition is that validation succeeds when the software function in a manner that can reasonably expected by the customer. After validation test has been conducted one of two possible conditions exists.

One is the function or performance characteristics confirm to specifications and are accepted and the other is deviation from specification is uncovered and a deficiency list is created. Proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

* **Output Testing**

After performing validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated by the system under consideration. Here the output format is considered in two ways, one is on the screen and other is the printed format. The output format on the screen is found to be correct as the format was designed in the system designed phase according to the user needs.

For the hard copy also the output comes as the specified requirements by the users. Hence output testing does not result any corrections in the system.

* **User Acceptance Testing**

User acceptance of a system is the key factor of the success of any system. The system under study is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required.

**ADVANTAGES OF PROJECT**

* User can search for doctor’s help at any point of time.
* User can talk about their Heart Disease and get instant diagnosis.
* Doctors get more clients online.
* Random forest is one of the most accurate classifier algorithms among decision tree.
* Very useful in case of emergency.

**Disadvantages:**

The system is not fully automated, it needs data from user for full diagnosis.

**Application:**

This application can be used by all patients or their family members who need help in emergency.

**Web Application is:**

1. Load Balancing:

Since the system will be available only the admin logs in the amount of load on server will be limited to time period of admin access.

1. Easy Accessibility:

Records can be easily accessed and store and other information respectively.

1. User Friendly:

The web application will be giving a very user friendly approach for all user.

1. Efficient and reliable:

Maintaining the all secured and database on the server which will be accessible according the user requirement without any maintenance cost will be a very efficient as compared to storing all the customer data on the spreadsheet or in physically in the record books.

1. Easy maintenance:

Heart Disease Prediction is design as easy way. So maintenance is also easy.

**CONCLUSION**

The Heart Disease Prediction, historically viewed as a necessary burden in medical offices, healthcare facilities and wellness centers, can be completely automated through an inefficient online software program. The benefits of implementing this technology touch everyone involved in the scheduling process, as administrators and users can conduct their tasks more efficiently and accurately. The system extracts hidden knowledge from a historical heart disease database. This system can be further enhanced and expanded.

**BIBLIOGRAPHY**

* **Websites**
* en.wikipedia.org
* Microsoft Developer Network (MSDN): http://msdn2.microsoft.com/en-us/default.aspx: This is a valuable online resource, and is a must for any developer using Microsoft tools.
* http://www.asp.net/: This is the official Microsoft ASP.NET web site. It has a lot of: tutorials, training videos, and sample projects.
* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.300.319&rep=rep1&type=pdf