**Food ordering system**

**OBJECTIVE:**

To design a food ordering system that allows students to order their food from college cafeteria using an app, to avoid crowd and allows online payment without fraud.

**ABSTRACT:**

Applications in these industries are becoming more important than ever thanks to the widespread use of different mobile operating systems Android. As each of these platforms needs a unique set of programming skills, it takes a lot of time and money for developers to create apps for each of these platforms separately. Web-based solutions can therefore be utilized to reduce these issues. Platform independence can be attained through cross-platform development since users can write their code in a single language that is easily translated to a variety of systems. The system will helpful to the user to reduce the waiting time in the café for food in the college. The user will order the food based on his interest the user will get the message complete his order. The order gets placed to the cafeteria only after the payment is made. Once the cafeteria receives the order, they can start the preparation of the dish. After the dish is ready, the student gets a message to their registered phone number to collect the food item. Student is required to show their order id to the cafeteria staff to collect their assigned order.

**INTRODUCTION:**

It is known globally that, in today’s market, it is extremely difficult to start a new small-scale business and live-through the competition from the well-established and settled owners. In fast paced time of today, when everyone is squeezed for time, the majority of people are finicky when it comes to placing a food order. The customers of today are not only attracted because placing an order online is very convenient but also because they have visibility into the items offered, greatly simplifies the ordering process for both the customer and the restaurant. System presents an interactive and up-to-date menu with all available options in an easy-to-use manner. Customer can choose one or more items to place an order which will land in the Cart. Customer can view all the order details in the cart before checking out. At the end, customer gets order confirmation details. Once the order is placed it is entered in the database and retrieved in pretty much real time. This allows Restaurant Employees to quickly go through the orders as they are received and process all orders efficiently and effectively with minimal delays and confusion.

**LITERATURE REVIEW:**

**Foody – Smart Restaurant Management and Ordering System**

**Vindya Liyanage, Achini Ekanayake, Hiranthi Premasiri, Prabhashi Munasinghe**

Customers play a vital role in the contemporary food industry when determining the quality of the restaurant and its food. Restaurants give considerable attention to customers’ feedback about their service, since the reputation of the business depends on it. Key factors of evaluating customer satisfaction are, being able to deliver the services effectively to lessen the time of consumption, as well as maintaining a high quality of service.

**Stimulus Factors of Order Online Food Delivery**

**Yakob Utama Chandra; Cadelina Cassandra**

Food order from mobile application is popular nowadays, this way can help customers order food effectively and faster. It also can offer maximum benefit for suppliers and customers. So, the customer can get the value of online food order delivery.

**Online Food Delivery Platforms and Restaurants**

**KEQUAN LI, YAN CHEN, JIACHEN LIU, LIN ZHANG AND XIANGWEI MU**

This study applied evolutionary game theory to examine the interaction mechanism of the problematic behaviors between OFD platforms and restaurants. By theoretical research and simulation, we analyzed the conditions under which the game can converge to the expected outcome. In addition, we discussed the measures that are most important to promote restaurants using degradable food packaging in their service.

**EXISTING SYSTEM:**

To physically visit the cafeteria for eating food, and have to make payment through cash mode most of the times due to unawareness of advanced technologies at certain places. In this method time as well as physical work is required, among which time is something that no one has in ample amount. The traditional food ordering procedure is not efficient enough for hotels and restaurant, as they have to deal with crowd, in their cafeteria.

**Disadvantages:**

* The missing of orders.
* The time line is more to order.

**PROPOSED SYSTEM**

The proposed system is more useful to the user to reduce the waiting time in the cafeteria. The user will choose his cafeteria stalls. The user will view the food items and place the order based on his interest. The user after payment need to add the transaction id for the conformation. The user will get the message once his order is completed. The cafeteria will ask the order id for the user before collecting.

**Advantages:**

* They will work faster.
* The uI is friendly for the user.

**SYSTEM REQUIREMENTS:**

* **H/W System Configuration: -**
* Processor - I3/Intel Processor
* RAM - 8 GB
* Hard Disk - 1TB

**S/W System Configuration: -**

* Operating System - Windows 10
* JDK - java
* Plugin -Kotlin
* SDK - Android
* IDE -Android studio
* Database` - SQL, php

**SOFTWARE INSTALLATION FOR ANDROID PROJECT:**

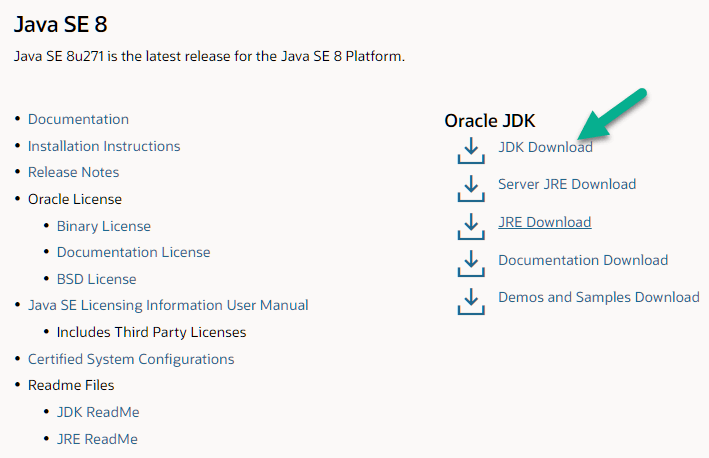
1. **Software Installation of JDK kit**

This Java Development Kit (JDK) allows you to code and run Java programs. It's possible that you install multiple JDK versions on the same PC. But it’s recommended that you install only latest version.

## How to install Java for Windows

Following are the steps for JDK 8 free download for 32 bit or JDK 8 download 64 bit and installation

**Step 1)** Go to [link](https://www.oracle.com/java/technologies/javase-downloads.html) Click on JDK Download for Java



**Step 2)** Next,

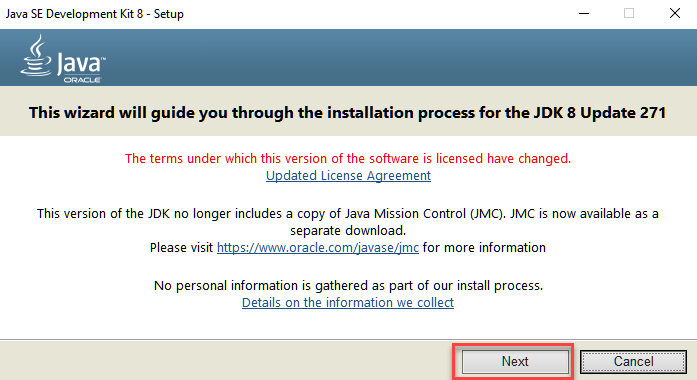
1. Accept License Agreement
2. Download Java 8 JDK for your version 32 bit or JDK 8 download for windows 10 64 bit.



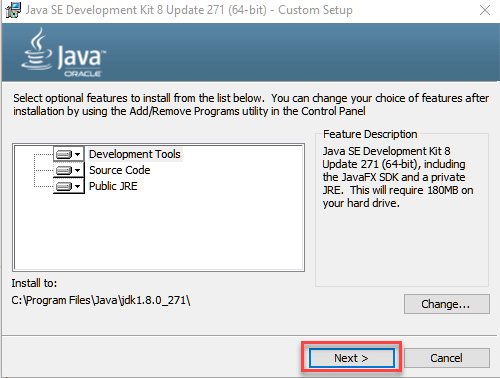
**Step 3)** when you click on the Installation link the popup will be open. Click on I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE and you will be redirected to the login page. If you don't have an oracle account you can easily sign up by adding basics details of yours.



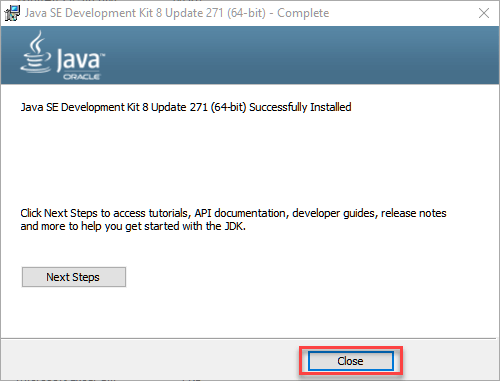
**Step 4)** once the Java JDK 8 download is complete, run the exe for install JDK. Click Next



**Step 5)** Select the PATH to install Java in Windows and click next.



**Step 6)** Once you install Java in windows, click Close



## How to set Environment Variables in Java: Path and Class path

The PATH variable gives the location of executable like javac, java etc. It is possible to run a program without specifying the PATH but you will need to give full path of executable like **C:\Program Files\Java\jdk-13.0.1\bin\javac A.java** instead of simple **javac A.java**

The CLASSPATH variable gives location of the Library Files.

Let's look into the steps to set the PATH and CLASSPATH

**Step 1)** Right Click on the My Computer and Select the properties

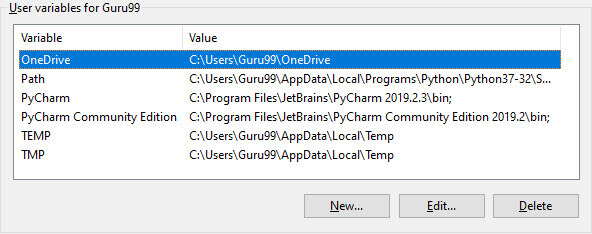


**Step 2)** Click on advanced system settings

**Step 3)** Click on Environment Variables



**Step 4)** Click on new Button of User variables



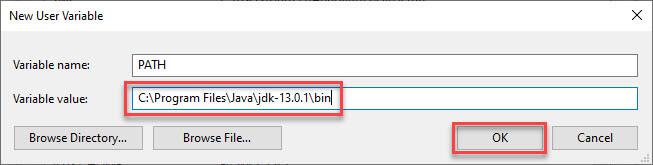
**Step 5)** Type PATH in the Variable name.



**Step 6)** Copy the path of bin folder which is installed in JDK folder.



**Step 7)** Paste Path of bin folder in Variable value and click on OK Button.



**Note:** In case you already have a PATH variable created in your PC, edit the PATH variable to

PATH = <JDK installation directory>\bin; %PATH%;

Here, %PATH% appends the existing path variable to our new value

**Step 8)**You can follow a similar process to set CLASSPATH.



**Note:** In case you java installation does not work after installation, change classpath to

CLASSPATH = <JDK installation directory>\lib\tools.jar;

**Step 9)** Click on OK button



**Step 10)** Go to command prompt and type java commands.

If you see a screen like below, Java is installed.



**2.Android Studio IDE and SDK Installation**

Installing Android software is probably the most challenging part of this project. It takes times - from 30 minutes to *n* hours to forever - depending on your luck, your programming knowledge, and your PC. You probably need a fairly decent PC (with 8GB RAM) and 10GB of free disk space to run the Android emulator!!! Running on "actual" Android phone/tablet requires much lesser resources.

##### **Step 0: Pre-Installation Check List**

1. Before installing Android SDK, you need to install Java Development Kit (JDK). Read "[How to install JDK](https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_HowTo.html)". Ensure that your JDK is at or above 1.8. You can check your JDK version with command "javac -version".
2. Uninstall older version(s) of "Android Studio" and "Android SDK", if any.
3. The installation and many operations take a LONG time to complete. Do NOT stare at your screen or at the ceiling. Browse through the "Android Developers" @ [https://developer.android.com](https://developer.android.com/index.html). For developers, check out the "Developer Guides".
4. We need to install two huge packages:
   1. Android Studio (IDE) (about 1 GB), which is an Integrated Development Environment (IDE) based on IntelliJ (a popular Java IDE); and
   2. Android SDK (Software Development Kit) (about 5 GB) for developing and testing Android apps.

##### **Step 1: Install "Android Studio IDE"**

Reference: "Install Android Studio" @ <https://developer.android.com/studio/install>.

##### (For Windows)

1. Check that environment variable JAVA\_HOME is set to the JDK installation directory via command "set JAVA\_HOME". Otherwise, Follow the steps [HERE](https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_HowTo.html#Set-JAVA_HOME).
2. Check the system requirements for Android Studio/SDK @ <https://developer.android.com/studio#Requirements> e.g., For Windows 10, recommended 8GB of RAM, 4GB of disk space, and 1280x800 minimum screen resolution.
3. Go to "Android Studio" @ <https://developer.android.com/studio> ⇒ Click "Download Android Studio 3.6.x for Windows 64-bit (749MB)", e.g., "android-studio-ide-192.xxxxxxx-windows.exe".
4. Run the downloaded installer ⇒ You may watch a short video @ <https://developer.android.com/studio/install>.
   1. In "Choose Components", select "Android Studio" and "Android Virtual Device".
   2. In "Configuration Settings Install Location", accept the default "C:\Program Files\Android\Android Studio".
   3. In "Choose Start Menu Folder", accept the default ⇒ Install.
   4. Launch Android Studio. Continue to the next Step.

by default, the "Android Studio IDE" will be installed in "C:\Program Files\Android\Android Studio", and the "Android SDK" in "c:\Users\username\AppData\Local\Android\Sdk".

Notes: You can also use the ZIP version: Download the Windows 64-bit ZIP version (about 1.5GB) ⇒ UNZIP into a folder of your choice ⇒ Run "bin\studio64.exe" to launch the Android Studio ⇒ It will enter the "setup" for the first launch ⇒ "Do not Import Settings" ⇒ In "Welcome", click "Next" ⇒ In "Install Type", choose "Custom" (so that you can see what is going on) ⇒ In "Select Default JDK Location", use default ⇒ In "Select UI Theme", choose one that you like ⇒ In "SDK Components Setup", select "Android Virtual Device (1.05GB) ⇒ Take note of the "Android SDK Location" with default of "C:\Users\username\AppData\Local\Android\Sdk" ⇒ In "Emulator Settings", use default ⇒ In "Verify Settings", check the settings and choose "Finish" ⇒ In "Download Components", click "Details" and check that nothing fails ⇒ Wait ⇒ Wait ⇒ Wait.

##### **Step 2: Installing Android SDK**

##### This step takes a long time as you need to download about 3GB of zip data, and expand to 5 GB of disk data, even for the minimum configuration.

Note: You can actually copy the SDK from another computer with the same OS.

##### (For Windows and macOS)

1. Launch Android Studio ⇒ It will run the "setup" wizard for the first launch.
   1. Choose "do not import previous settings".
   2. In "Welcome", choose "next".
   3. In "Install Type", choose "Standard" (default).
   4. In "Select UI Theme", choose one that you like (or default).
   5. In "SDK Components Setup", make sure that "Android Virtual Device" is selected and take note of the SDK directory (by default @ c:\Users\*username*\AppData\Local\Android\Sdk for Windows, "~/Library/Android/sdk" for macOS) ⇒ Finish ⇒ Wait ⇒ Wait ⇒ Wait ⇒ Wait ⇒ Wait ⇒ Wait.
2. (For Windows) Use "File Explorer" to check the SDK installed directory. Take note that the "App Data" is a hidden directory. You need to choose "View" menu ⇒ Uncheck "Hidden Items" to see this directory.  
   (For macOS) Use "Finder" to check the SDK installed directory.
3. Also use "Android Studio" to check the SDK packages installed by selecting "Configure" (at the bottom of Android Studio) ⇒ "SDK Manager" ⇒ "Android SDK" (sidebar):
   1. Under "SDK Platforms" tab:
      * Android 10.0 (Q) (API Level 29)
   2. Under "SDK Tools" tab:
      * Android SDK Build Tools 30-rc1
      * Android Emulator (30.0.0)
      * Android SDK Platform-Tools (29.0.6)
      * Intel x86 Emulator Accelerator (HAXM installer) (7.5.6)

**STEPS FOR EXECUTING THE PROJECTS**

**Step 1:**

Open Android Studio

**Step2:**

Choose a virtual device or Physical device from the menu

**Step3:**

Click on the project Run

**Step4:**

View the application performance on virtual or Physical device.

**IMPLEMENTATION**

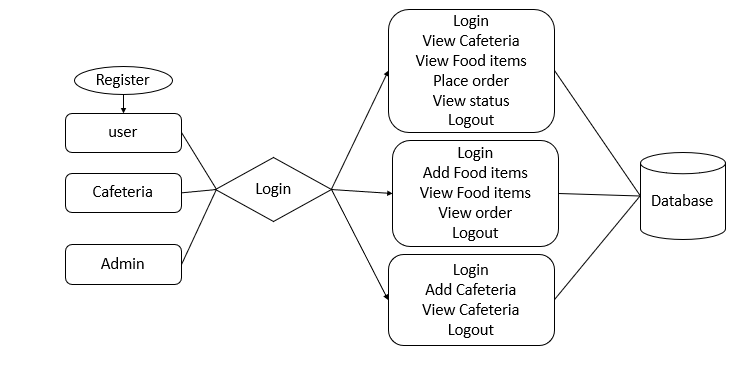
**Module And Functionalities:**

**Admin:** the admin will login the add the cafeteria and view the cafeteria.

**Cafeteria:** the cafeteria login the cafeteria will add the items in the cafeteria. The cafeteria will view the order and send the completed status to the user in message.

**User:** the using register first after the login with his email and password. the user views the cafeteria and they will view the food items and place the order the user will pay the payment and add the transaction id for the cafeteria for the payment conformation.

**Architecture:**



**SYSTEM DESIGN**

**UML DIAGRAMS**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



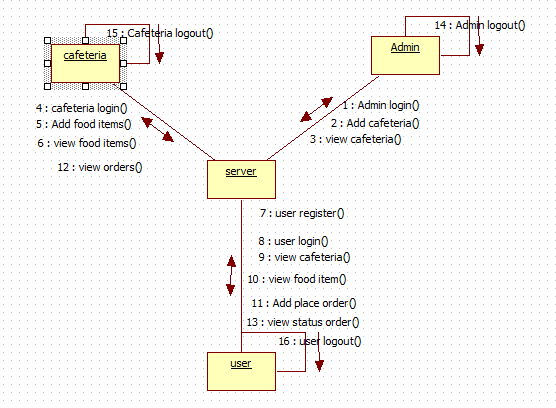
**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**COLLABORATION DIAGRAM:**

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

****

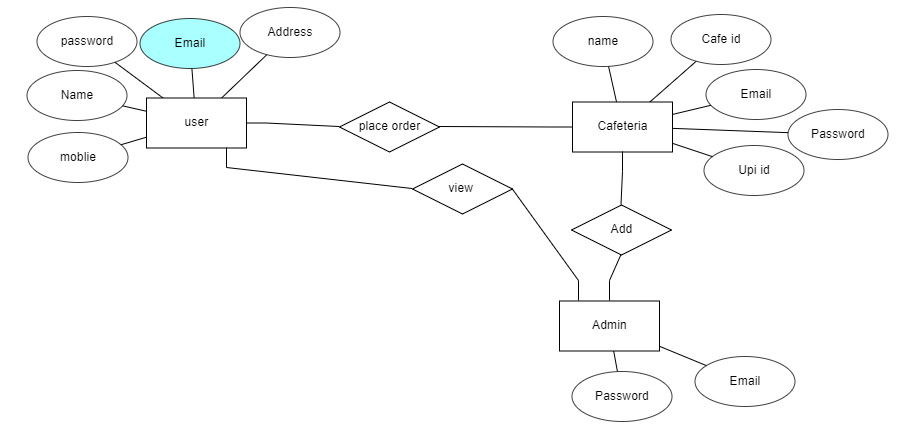
**Component Diagram:**

****

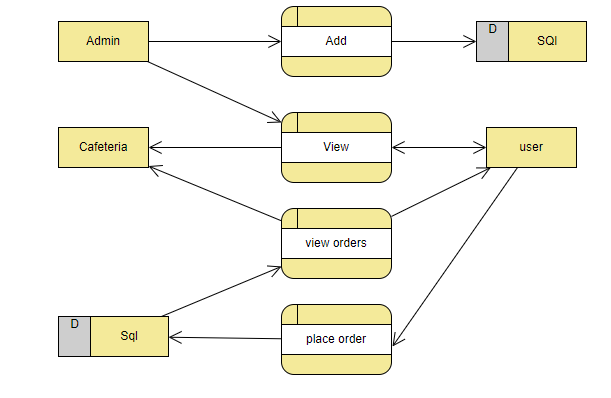
**Deployment Diagram:**



**ER Diagram**:



**DFD Diagram:**



**Data Dictionary:**

A data dictionary contains metadata i.e., data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

The data dictionary in general contains information about the following

* Names of all the database tables and their schemas.
* Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
* Physical information about the tables such as where they are stored and how.
* Table constraints such as primary key attributes, foreign key information etc.
* Information about the database views that are visible.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data type** | **Variable size** | **Key** | **Data** |
| **id** | **Int** | **20** | **Primary key** | **1** |
| **Name** | **varchar** | **20** | **NULL** | **test** |
| **email** | **varchar** | **30** | **NULL** | **rest@gmail.com** |
| **password** | **varchar** | **30** | **NULL** | **Test123** |
| **mobile** | **number** | **10** | **NULL** | **9999999999** |

**SOFTWARE DEVELOPMENT LIFE CYCLE**

The meaning of Agile is swift or versatile. “Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance. Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Actually, Agile model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves. A few Agile SDLC models are given below: Crystal A tern Feature-driven development Scrum Extreme programming (XP) Lean development Unified process In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed.

The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed and deployed to the customers. Long-term plans are not made.

Agile model is the combination of iterative and incremental process models. Steps involve in agile SDLC models are:

* Requirement gathering
* Requirement Analysis
* Design Coding
* Unit testing
* Acceptance testing

The time to complete an iteration is known as a Time Box. Time-box refers to the maximum amount of time needed to deliver an iteration to customers. So, the end date for an iteration does not change. Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time. The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.



**Principles of Agile model:**

* To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
* Agile model relies on working software deployment rather than comprehensive documentation.
* Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks.
* Requirement change requests from the customer are encouraged and efficiently incorporated.
* It emphasizes on having efficient team members and enhancing communications among them is given more importance. It is realized that enhanced communication among the development team members can be achieved through face-to-face communication rather than through the exchange of formal documents.
* It is recommended that the development team size should be kept small (5 to 9 people) to help the team members meaningfully engage in face-to-face communication and have collaborative work environment.
* Agile development process usually deploys Pair Programming. In Pair programming, two programmers work together at one work-station. One does code while the other reviews the code as it is typed in. The two programmers switch their roles every hour or so.

**Advantages:**

* Working through Pair programming produce well written compact programs which has fewer errors as compared to programmers working alone.
* It reduces total development time of the whole project. Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

**Disadvantages:**

* Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
* Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

**SOFTWARE ENVIRONMENT**

**Software Environment**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005.

Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release.

The Android Open-Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform.

The Android SDK provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android.

Features: -

* **Application framework** enabling reuse and replacement of components
* **Dalvik virtual machine** optimized for mobile devices
* **Integrated browser** based on the open-source Web Kit engine
* **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
* **SQLite** for structured data storage
* **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
* **GSM Telephony** (hardware dependent)
* **Bluetooth, EDGE, 3G, and WIFI** (hardware dependent)
* **Camera, GPS, compass, and accelerometer** (hardware dependent)
* **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

## Android Architecture



## Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

* **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
* **Media Libraries** - based on Packet Video’s Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
* **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
* **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
* **SGL** - the underlying 2D graphics engine
* **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
* **Free Type** - bitmap and vector font rendering
* **SQLite** - a powerful and lightweight relational database engine available to all applications

## Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the. dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

## Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

The Linux kernel is an operating system kernel used by the Linux family of Unix-like operating systems. It is one of the most prominent examples of free and open-source software.

The Linux kernel is released under the GNU General Public License version 2 (GPLv2), (plus some firmware images with various licenses), and is developed by contributors worldwide. Day-to-day development takes place on the Linux kernel mailing list.

The Linux kernel was initially conceived and created by Finnish computer science student Linus Torvalds in 1991. Linux rapidly accumulated developers and users who adapted code from other free software projects for use with the new operating system. The Linux kernel has received contributions from thousands of programmers.[10] Many Linux distributions have been released based upon the Linux kernel.

The Linux kernel has extensive support for and runs on many virtual machine architectures both as the host operating system and as a guest operating system. The virtual machines usually emulate Intel x86 family of processors, though in a few cases PowerPC or ARM processors are also emulated. At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the premise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.[28][29][30] Speculation about Google's intention to enter the mobile communications market continued to build through December 2006. Reports from the BBC and The Wall Street Journal noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded handset. Some speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

## Hardware running Android

The main supported platform for Android is the ARM architecture.

The Android OS can be used as an operating system for cellphones, netbooks and tablets, including the Dell Streak, Samsung Galaxy Tab, TV and other devices.[68][69] The first commercially available phone to run the Android operating system was the HTC Dream, released on 22 October 2008.[70] In early 2010 Google collaborated with HTC to launch its flagship [71] Android device, the Nexus One. This was followed later in 2010 with the Samsung-made Nexus.

The early feedback on developing applications for the Android platform was mixed. Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on 18 January 2008.) In December 2007, Merge Lab mobile startup founder Adam Macbeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time." Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the Snake game The Android Dev Phone is a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer, devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device.  
The Android software development kit (SDK) includes a comprehensive set of development tools.[80] These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. The SDK is downloadable on the android developer website. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.9 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse (currently 3.5 or 3.6) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).[81]

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible to root user only for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

### Android Operation System

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g., a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is led by Google.

Android uses a special virtual machine, e.g., the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore, you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for Eclipse . The ADT performs automatically the conversion from class to dex files and creates the apk during deployment.

Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a SQLite database.

Every Android applications run in its own process and under its own user id which is generated automatically by the Android system during deployment. Therefore, the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

### Important Android components

An Android application consists out of the following parts:

* Activity - Represents the presentation layer of an Android application, e.g., a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
* Views - The User interface of an Activities is built with widgets classes which inherent from "android. view. View". The layout of the views is managed by "android. view. View Groups".
* Services - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
* Content Provider - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
* Intents are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or asked the Android system for registered services and applications for an intent (implicit intents). For example, the application could ask via an intent for a contact application. Application registers themself to an intent via an Intent Filter. Intents are a powerful concept as they allow to create loosely coupled applications.
* Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.
* A Java Virtual Machine (JVM) enables a set of computer software programs and data structures to use a virtual machine model for the execution of other computer programs and scripts. The model used by a JVM accepts a form of computer intermediate language commonly referred to as Java bytecode. This language conceptually represents the instruction set of a stack-oriented, capability architecture. Sun Microsystems states there are over 4.5 billion JVM-enabled devices
* A JVM can also execute bytecode compiled from programming languages other than Java. For example, Ada source code can be compiled to execute on a JVM. JVMs can also be released by other companies besides Oracle (the developer of Java) — JVMs using the "Java" trademark may be developed by other companies as long as they adhere to the JVM specification published by Oracle and to related contractual obligations.
* Java was conceived with the concept of WORA: "write once, run anywhere". This is done using the Java Virtual Machine. The JVM is the environment in which java programs execute. It is software that is implemented on non-virtual hardware and on standard operating systems.
* JVM is a crucial component of the Java platform, and because JVMs are available for many hardware and software platforms, Java can be both middleware and a platform in its own right, [clarification needed] hence the trademark write once, run anywhere. The use of the same bytecode for all platforms allows Java to be described as "compile once, run anywhere", as opposed to "write once, compile anywhere", which describes cross-platform compiled languages. A JVM also enables such features as automated exception handling, which provides "root-cause" debugging information for every software error (exception), independent of the source code.
* A JVM is distributed along with a set of standard class libraries that implement the Java application programming interface (API). Appropriate APIs bundled together form the Java Runtime Environment (JRE).
* Java's execution environment is termed the Java Runtime Environment, or JRE.
* Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of .class files. A program may consist of many classes in different files. For easier distribution of large programs, multiple class files may be packaged together in a .jar file (short for Java archive).
* The Java application launcher, java, offers a standard way of executing Java code. Compare javaw.[2]
* The JVM runtime executes .class or .jar files, emulating the JVM instruction set by interpreting it, or using a just-in-time compiler (JIT) such as Oracle's Hotspot. JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed. There are also ahead-of-time compilers that enable developers to precompile class files into native code for particular platforms.
* Like most virtual machines, the Java Virtual Machine has a stack-based architecture akin to a microcontroller/microprocessor. However, the JVM also has low-level support for Java-like classes and methods, which amounts to a highly idiosyncratic [clarification needed] memory model and capability-based architecture.

Download the Android SDK

Welcome Developers! If you are new to the Android SDK, please read the steps below, for an overview of how to set up the SDK.

If you're already using the Android SDK, you should update to the latest tools or platform using the Android SDK and AVD Manager, rather than downloading a new SDK starter package. See Adding SDK Components.

is now downloading. Follow the steps below to get started.

Here an overview of the steps you must follow to set up the Android SDK:

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from the table above. (If you're on Windows, download the installer for help with the initial setup.)
3. Install the ADT Plugin for Eclipse (if you'll be developing in Eclipse).
4. Add Android platforms and other components to your SDK.
5. Explore the contents of the Android SDK (optional).

To get started, download the appropriate package from the table above, then read the guide to Installing the SDK.

# Installing the SDK

## Step 1. Preparing Your Development Computer

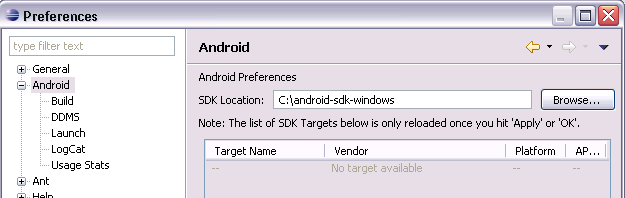
Before getting started with the Android SDK, take a moment to confirm that your development computer meets the System Requirements. In particular, you might need to install the JDK, if you don't have it already.

If you will be developing in Eclipse with the Android Development Tools (ADT) Plugin—the recommended path if you are new to Android—make sure that you have a suitable version of Eclipse installed on your computer as described in the System Requirements document. If you need to install Eclipse, you can download it from this location: The "Eclipse Classic" version is recommended. Otherwise, a Java or RCP version of Eclipse is recommended.

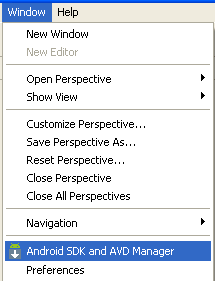
Use the Eclipse update manager to install all available plugins for the Android Development Tools (ADT) from the URL https://dl-ssl.google.com/android/eclipse/.

### Configuration

In Eclipse open the Preferences dialog via Windows -> Preferences. Select Android and maintain the installation path of the Android SDK.



Select Window -> Android SDK and AVD Manager from the menu.



Select available packages and select the latest version of the SDK.

Step 2. Downloading the SDK Starter Package

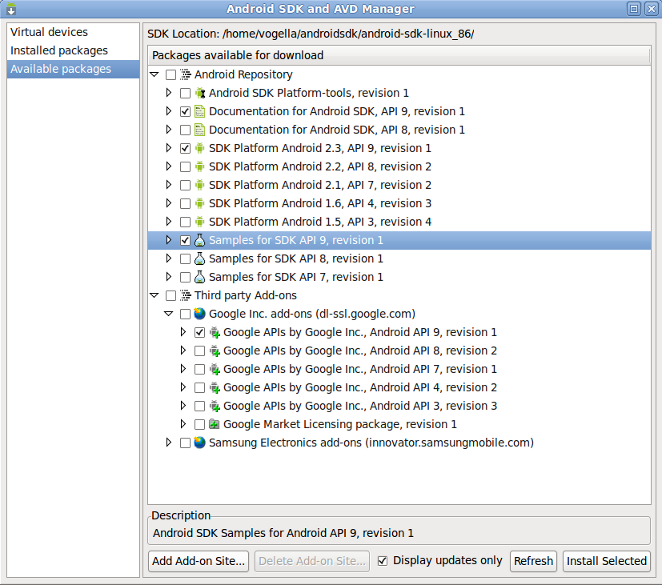
The SDK starter package is not a full development environment—it includes only the core SDK Tools, which you can use to download the rest of the SDK components (such as the latest Android platform).

If you haven't already, get the latest version of the SDK starter package from the SDK download page.

If you downloaded a .zip or .tgz package (instead of the SDK installer), unpack it to a safe location on your machine. By default, the SDK files are unpacked into a directory named android-sdk-<machine-platform>.

If you downloaded the Windows installer (.exe file), run it now and it will check whether the proper Java SE Development Kit (JDK) is installed (installing it, if necessary), then install the SDK Tools into a default location (which you can modify).

Make a note of the name and location of the SDK directory on your system—you will need to refer to the SDK directory later, when setting up the ADT plugin and when using the SDK tools from the command line.



## Step 3. Installing the ADT Plugin for Eclipse

Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT), that is designed to give you a powerful, integrated environment in which to build Android applications. It extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, debug your applications using the Android SDK tools, and even export signed (or unsigned) APKs in order to distribute your application. In general, developing in Eclipse with ADT is a highly recommended approach and is the fastest way to get started with Android.

If you'd like to use ADT for developing Android applications, install it now. Read Installing the ADT Plugin for step-by-step installation instructions, then return here to continue the last step in setting up your Android SDK.

If you prefer to work in a different IDE, you do not need to install Eclipse or ADT. Instead, you can directly use the SDK tools to build and debug your application. The Introduction to Android application development outlines the major steps that you need to complete when developing in Eclipse or other IDEs.





## Step 4. Adding Platforms and Other Components

The last step in setting up your SDK is using the Android SDK and AVD Manager (a tool included in the SDK starter package) to download essential SDK components into your development environment.

The SDK uses a modular structure that separates the major parts of the SDK—Android platform versions, add-ons, tools, samples, and documentation—into a set of separately installable components. The SDK starter package, which you've already downloaded, includes only a single component: the latest version of the SDK Tools. To develop an Android application, you also need to download at least one Android platform and the associated platform tools. You can add other components and platforms as well, which is highly recommended.





If you used the Windows installer, when you complete the installation wizard, it will launch the Android SDK and AVD Manager with a default set of platforms and other components selected for you to install. Simply click **Install** to accept the recommended set of components and install them. You can then skip to Step 5, but we recommend you first read the section about the Available Components to better understand the components available from the Android SDK and AVD Manager.

You can launch the Android SDK and AVD Manager in one of the following ways:

* From within Eclipse, select **Window > Android SDK and AVD Manager**.
* On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.
* On Mac or Linux, open a terminal and navigate to the tools/ directory in the Android SDK, then execute:

To download components, use the graphical UI of the Android SDK and AVD Manager to browse the SDK repository and select new or updated components (see figure 1). The Android SDK and AVD Manager installs the selected components in your SDK environment. For information about which components you should download, see Recommended Components.

The Android Repository offers these types of components:

* **SDK Tools** — Contains tools for debugging and testing your application and other utility tools. These tools are installed with the Android SDK starter package and receive periodic updates. You can access these tools in the <sdk>/tools/ directory of your SDK. To learn more about them, see SDK Tools in the developer guide.
* **SDK Platform-tools** — Contains platform-dependent tools for developing and debugging your application. These tools support the latest features of the Android and platform are typically updated only when a new platform becomes available. You can access these tools in the <sdk>/platform-tools/ directory. To learn more about them, see Platform Tools in the developer guide.
* **Android platforms** — An SDK platform is available for every production Android platform deployable to Android-powered devices. Each SDK platform component includes a fully compliant Android library, system image, sample code, and emulator skins. To learn more about a specific platform, see the list of platforms that appears under the section "Downloadable SDK Components" on the left part of this page.
* **USB Driver for Windows** (Windows only) — Contains driver files that you can install on your windows computer, so that you can run and debug your applications on an actual device. You do not need the USB driver unless you plan to debug your application on an actual Android-powered device. If you develop on Mac OS X or Linux, you do not need a special driver to debug your application on an Android-powered device. See Using Hardware Devices for more information about developing on a real device.
* **Samples** — Contains the sample code and apps available for each Android development platform. If you are just getting started with Android development, make sure to download the samples to your SDK.
* **Documentation** — Contains a local copy of the latest multi-version documentation for the Android framework API.

The Third-party Add-ons provide components that allow you to create a development environment using a specific Android external library (such as the Google Maps library) or a customized (but fully compliant) Android system image. You can add additional Add-on repositories by clicking **Add Add-on Site**.

ECLIPSE:

Eclipse is an open-source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle. Many people know us, and hopefully love us, as a Java IDE but Eclipse is much more than a Java IDE.

The Eclipse open-source community has over 60 open-source projects. These projects can be conceptually organized into seven different "pillars" or categories:

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)

The Eclipse community is also supported by a large and vibrant ecosystem of major IT solution providers, innovative start-ups, universities and research institutions and individuals that extend, support and complement the Eclipse Platform.

The exciting thing about Eclipse is many people are using Eclipse in ways that we have never imagined. The common thread is that they are building innovative, industrial strength software and want to use great tools, frameworks and runtimes to make their job easier.

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, Ruby (including Ruby on Rails framework), Scala, Clojure, and Scheme. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

ARCHITECTURE:

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The runtime system of Eclipse is based on Equinox, an OSGi standard compliant implementation.

This plug-in mechanism is a lightweight software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX,[2] networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with Subversion support provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java Development Tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.

## Rich Client Platform

* Equinox OSGi – a standard bundling framework.
* Core platform – boot Eclipse, run plug-ins
* Standard Widget Toolkit (SWT) – a portable widget toolkit
* JFace – viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors
* Eclipse Workbench – views, editors, perspectives, wizards

## History

* Eclipse began as an IBM Canada project. It was developed by Object Technology International (OTI) as a Java-based replacement for the Smalltalk based Visual Age family of IDE products,[4] which itself had been developed by OTI.[1] In November 2001, a consortium was formed to further the development of Eclipse as open source. In January 2004, the Eclipse Foundation was created.[5]

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture.[6]

Eclipse was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL).[7] Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.[8]

According to Lee Nackman, Chief Technology Officer of IBM's Rational division at that time and later head of Rational software development and support, the name "Eclipse" was chosen to target Microsoft's Visual Studio product, and not Sun Microsystems.[9] Ironically, Nackman is now himself a Microsoft employee.

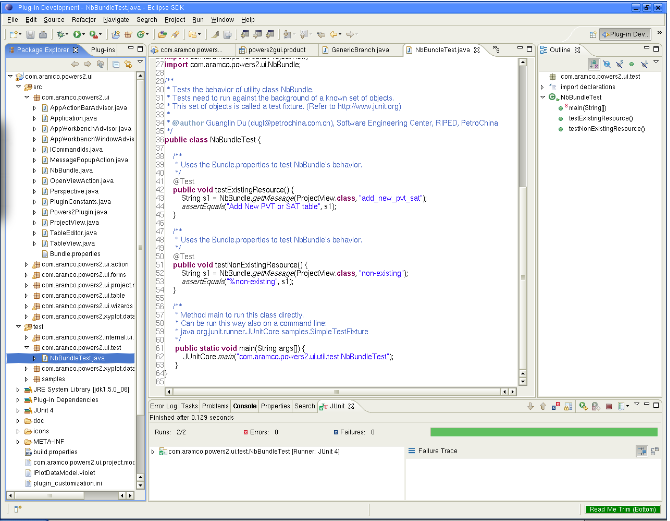
# Eclipse (SDK)

Eclipse Software Development Kit (SDK) is a Java based open-source integrated development environment (IDE) which combines a number of different Eclipse projects including Platform, Java Development Tools (JDT) and the Plug-in Development Environment (PDE).

Eclipse can be used to create a large array of software applications using languages ranging from PHP, C++ programs, to Java. It is one of the most popular development tools in both the open-source and commercial worlds.

It provides Java editing with validation, incremental compilation, cross-referencing, code assist; an XML Editor; Mylyn; and much more.

Eclipse is released under the Eclipse Foundation, a commercially friendly license that allows organizations to include Eclipse software in their commercial products, while at the same time asking those who create derivative works of EPL code to contribute back to the community.



Eclipse Platform

The Eclipse Platform provides the core frameworks and services upon which all plug-in extensions are created. It also provides the runtime in which plug-ins are loaded, integrated, and executed. The primary purpose of the Platform is to enable other tool developers to easily build and deliver integrated tools.

Features include:

* Supports the construction of a variety of tools for application development
* Supports an unrestricted set of tool providers, including independent software vendors (ISVs)
* Supports tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF)
* Facilitates seamless integration of tools within and across different content types and tool providers
* Supports both GUI and non-GUI-based application development environments

Java Development Tools (JDT)

The JDT project provides the tool plug-ins that implement a Java IDE supporting the development of any Java application, including Eclipse plug-ins. It adds a Java project nature and Java perspective to the Eclipse Workbench as well as a number of views, editors, wizards, builders, and code merging and refactoring tools. The JDT project allows Eclipse to be a development environment for itself.

Features include:

* Java projects with source files arranged in package directories
* Editing with keyword and syntax colouring, outline showing declaration structure
* Code formatter
* Refactoring
* Search
* Compare
* Compile - JCK-compliant Java compiler
* Run Java programs in a separate target Java virtual machine
* Debug programs with JPDA-compliant Java virtual machine

### Android Source Code

The following step is optional.

During Android development it is very useful to have the Android source code available as Android uses a lot of defaults.

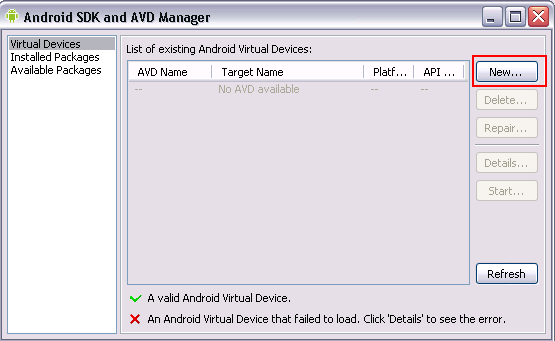
Haris Peco maintains plugins with provides access to the Android Source code. Use the Eclipse update manager to install two of his plugins.

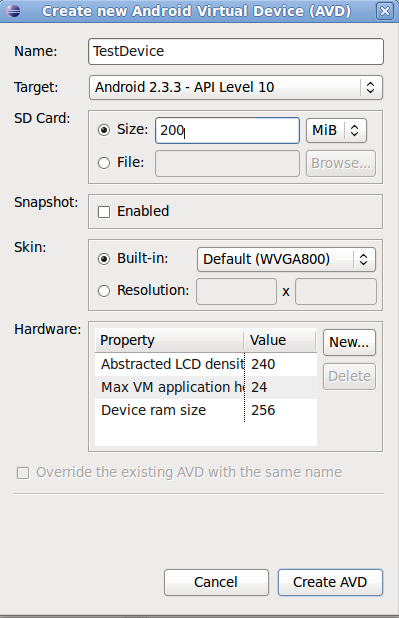
## Create an Android Emulator Device

The Android tools include an emulator. This emulator behaves like a real Android device in most cases and allow you to test your application without having a real device. You can emulate one or several devices with different configurations. Each configuration is defined via an "Android Virtual Device" (AVD).

To define an AVD press the device manager button, press "New" and maintain the following.







Press "Create AVD". This will create the device and display it under the "Virtual devices". To test if your setup is correct, select your device and press "Start".



## 4. Error handling

Things are not always working as they should be. Several users report that get the following errors:

1. Project ... is missing required source folder: 'gen'
2. The project could not be built until build path errors are resolved.
3. Unable to open class file R.java.

To solve this error, select from the menu Project -> Clean.

If you having problems with your own code, you can use the Log Cat viewer as described in Log Cat Viewer.

**SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**INTEGRATION TESTING**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points

**WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**UNIT TESTING:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# INTEGRATION TESTINGSS

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Conclusion:**

Even though this research focusses on the college cafeteria, the team will be focusing on reducing time wastage because customers can check availability of tables before entering the cafeteria. The user will view the food item are available in cafeteria the user will place his order based on his interest the user will pay the payment in the application itself after the payment the cafeteria will get the payment is paid. Once the order will complete the cafeteria will send a message to the user. The user once collected by showing the order id to the cafeteria.

**TESTING RESULTS**

**TESTING CASES**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test case id | Test Scenario | Test Steps | Prerequisites | Test Data | Expected result | Actual result | Test status |
| **#CVD001** | To authenticate a successful signup with user data | * User navigate the signup page * Enter the valid user data * Click on signup button | User data | Username  Password  Mobile  Email  location | When the user submits the user data, data should be store in database successfully | As Expected, | Pass |
| **#CVD002** | To authenticate a successful login with user data | * User navigate the login page. * Enter the valid username, password. * Click on login button | Username, password | Username password | When the user submits the user data, data should be authenticated successfully | As Expected, | Pass |

**REFERENCES:**

* Vindya Liyanage, Achini Ekanayake, Hiranthi Premasiri, Prabhashi Munasinghe,"Foody – Smart Restaurant Management and Ordering System ",2018
* Yakob Utama Chandra; Cadelina Cassandra,"Stimulus Factors of Order Online Food Delivery",2019
* KEQUAN LI, YAN CHEN, JIACHEN LIU, LIN ZHANG, AND XIANGWEI MU,"Online Food Delivery Platforms and Restaurants",2021
* S. M. Hassan, "Top Online Shopping E-companies and their Strength and Weakness", 2014.
* Susan Stepney and P. Lord Stephen, "A Formal Specification of an Access Control System", 1987.
* V. A. Bharadi, Intelligent e-Restaurant using Android OS, Mumbai, 2013.
* M. Rajesh, "E-Restaurant: Online Restaurant Management", 2015.
* A. Patil, "SMART RESTAURANT SYSTEM USING ANDROID", 2017.
* M. Y. Ricky, "Mobile Food Ordering Application using Android OS Platform", 2014.
* M. Maimaiti, X. Zhao, M. Jia, Y. Ru, and S. Zhu, ‘‘How we eat determines what we become: Opportunities and challenges brought by food delivery industry in a changing world in China,’’. 2018.