

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi – 590 014



A Mobile Application Development Mini Project Report On

“COLLEGE CANTEEN”

Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering

In

Computer Science & Engineering

By

Mr. KRISHNAMURTHI R B (4MW18CS039)

Mr. NIKHIL G POOJARY (4MW18CS047)

Under the guidance of

MS SOUMYA

Asst. Prof, Dept. of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Shri Madhwa Vadiraja Institute of Technology & Management
Vishwothama Nagar, Bantakal – 574 115, Udupi District August,
2021

SHRI MADHWA VADIRAJA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(A Unit of Shri Sode Vadiraja Mutt Education Trust @, Udupi) Vishwothama

Nagar, BANTAKAL – 574 115, Udupi District, Karnataka, INDIA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the Computer Graphics Project Work entitled “**COLLEGE CANTEEN**” has been carried out by **MR. KRISHNAMURTHI BHAGAWATH (4MW18CS039) AND NIKHIL G POOJARY(4MW18CS047)**, bonafide student of Shri Madhwa Vadiraja Institute of Technology and Management, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi during the year 2020-21. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The Mobile Application Development Mini Project Report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

MS. SOUMYA

Project Guide,

Dept. of CSE

Dr. NAGARAJ BHAT

Associate Professor and

Head, Dept. of CSE

External Viva

Name of the Examiners:

Signature with Date

1.

2.

ABSTRACT

The Android mobile platform has developed from its first phone in October 2008 to being the most popular smart phone operating system in the world by 2012. The explosive growth of the platform has been a significant win for consumers with respect to competition and features.

The market has been booming in the past few years that, there are now over 1,195,932 applications on the Android market. Due to the wide usage, it is necessary to provide users with security applications to manage the data in their personal smart phones.

The purpose of this project is to develop a computerized and mobilized College Canteen Management system that can be used to revolutionize the traditional management system that currently implemented in majority of the food and beverage industry and colleges. The traditional system that using by most of the food and beverage industry is the traditional manual ordering system which means all works and procedures is recorded through manpower manual work and it consist of a huge amount of paper work that is not effective and efficiency. This causes the business to encounter trouble which regarding human error due to the huge amount of manpower manual work that operating in each business routine. Thus, this computerized and mobilized college canteen management system is designed to assist the business routine in term of having better management as well as easier to handle daily business operation.

Acknowledgements

It is our duty to acknowledge the help rendered to us by various persons in completing this Mobile Application Development Mini Project titled “**COLEGE CANTEEN**”

We are indebted to Prof. **Dr. Thirumaleshwara Bhat**, Principal, for their advice and suggestions at various stages of the work. We also extend our heartfelt gratitude to **Dr. Vasudeva** for his assistance.

We express our deepest gratitude and respect to our guide **Dr. NAGARAJ BHAT**, HOD, Computer Science and Engineering, for his valuable guidance and encouragement while doing this project work.

It is our privilege our sincerest regards to our project coordinator, **MS. Soumya**, for her valuable inputs, guidance, encouragement and whole-hearted cooperation throughout the duration of our project.

We extend our thanks to the Management of Shri Madhwa Vadiraja Institute of Technology and Management, Bantakal, Udupi for providing good laboratory and library facilities. We also remain grateful to the co-operation and help rendered by the teaching and non-teaching staff of the Computer Science and Engineering Department.

Lastly, we take this opportunity to offer our regards to all of those who have supported us directly or indirectly in the successful completion of this project work.

Thanking you all,
Krishnamurthi Bhagawath (4MW18CS039),
Nikhil G Poojary (4MW18CS047)

TABLE OF CONTENTS

CHAPTERS	PAGENO
1. INTRODUCTION	1
2. REQUIREMENT SPECIFICATION	8
3. DESIGN	9
4. IMPLEMENTATION	15
5. SNAPSHOTS	18
6. CONCLUSION	24
7. BIBLIOGRAPHY	26
8. USER GUIDE	27

INTRODUCTION

1.1 About android studio:

Android Studio is the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps. Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0. Since 7th May 2019, Kotlin is Google's preferred language for Android application development. Besides this, other programming languages are supported by Android Studio.

1.2 Features:

A specific feature of the Android Studio is an absence of the possibility to switch autosave feature off.

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

1.3 About Android libraries:

A module is a collection of source files and build settings that allow you to divide your project into discrete units of functionality. Project can have one or many modules and one module may use another module as a dependency. Each module can be independently built, tested, and debugged.

Additional modules are often useful when creating code libraries within your own project or when you want to create different sets of code and resources for different device types, such as phones and wearables, but keep all the files scoped within the same project and share some code.

You can add a new module to your project by clicking File > New > New Module.

1.3.1 Android app module:

Provides a container for your app's source code, resource files, and app level settings such as the module-level build file and Android Manifest file. When you create a new project, the default module name is "app".

In the Create New Module window, Android Studio offers the following types of app modules:

Phone & Tablet Module, Wear OS Module, Android TV Module, Glass Module

They each provide essential files and some code templates that are appropriate for the corresponding app or device type

1.3.2 Feature module:

Represents a modularized feature of your app that can take advantage of Play Feature Delivery. For example, with feature modules, you can provide your users with certain features of your app on-demand or as instant experiences through Google Play Instant.

1.3.3 Library module:

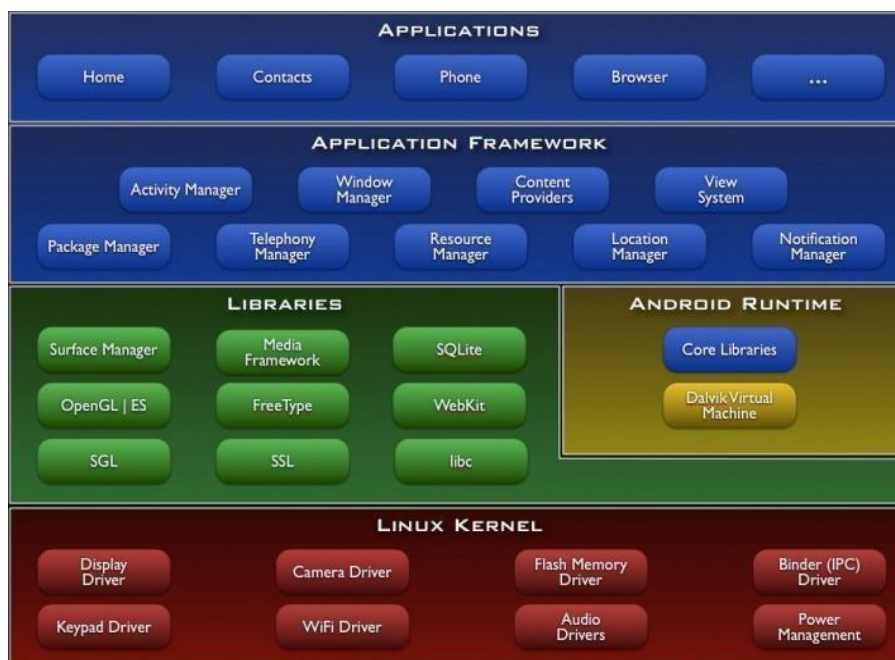
Provides a container for your reusable code, which you can use as a dependency in other app modules or import into other projects. Structurally, a library module is the same as an app module, but when built, it creates a code archive file instead of an APK, so it can't be installed on a device.

In the Create New Module window, Android Studio offers the following library modules:

- **Android Library:** This type of library can contain all file types supported in an Android project, including source code, resources, and manifest files. The build result is an Android Archive (AAR) file that you can add as a dependency for your Android app modules.
- **Java Library:** This type of library can contain only Java source files. The build result is a Java Archive (JAR) file that you can add as a dependency for your Android app modules or other Java projects

1.4 ARCHITECTURE OF ANDROID:

The following diagram shows the major components of the Android operating system. Each Section is described in more detail below.



1.4.1 Applications:

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

1.4.2 Application Framework:

The most important parts of the framework are as follows:

Activity Manager: This controls the life cycle of applications and maintains a common “back stack” for user navigation.

Content providers: These objects encapsulate data that needs to be shared between applications, such as contacts.

Resource manager: Resources are anything that goes with your program that is not code.

Location manager: An Android phone always knows where it is.

Notification manager: Events such as arriving messages, appointments, proximity alerts, alien invasions, and more can be presented in an unobtrusive fashion to the user.

View system: Enabling applications to access data from other applications or to share their own data

Telephony manager: Provides core telephoning functionalities

Window manager: The window manager creates display surfaces for the application. It is Responsible for organizing the screen and display of different layers of application

1.4.3 LIBRARIES:

The next layer above the kernel contains the Android native libraries. These shared libraries are all written in C or C++, compiled for the particular hardware architecture used by the phone, and preinstalled by the phone vendor.

Some of the most important native libraries include the following:

- **SURFACE MANAGER:** Manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
- **MEDIA LIBRARIES:** 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.

- **SQLite:** A powerful and lightweight relational database engine available to all applications
- **OPEN GL|ES:** Subset of the OpenGL 3D graphics API designed for embedded devices. It can use for hardware 3D acceleration.
- **FREE TYPE:** Bitmap and vector font rendering. It is used to rasterize the characters into bitmaps and provides into other font-related operations.
- **Web Kit:** A framework providing the basis for building a web browser based on the open-source Web Kit browser.
- **SGL:** SGL is the underlying 2D graphics engine.
- **SSL:** The Secure Socket Layer is a commonly-used protocol for managing the security of a message transmission on the internet.
- **Lib C:** A BSD – based implementation of the standard C library. Which is tuned for embedded Linux-based devices.

1.4.4 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.

Dalvik Virtual Machine:

The Dalvik virtual machine is simple Java interpreter machine, completely optimized for Android platform and which is developed to run on low-end memory mobile devices. One of the prominent aspects in Dalvik its capability to run along an application compilation enhancing the runtime performance of the applications. Dalvik is not exactly, a Java machine, because Dalvik could not read Java code, but consists its own byte code called “dex” and so the executable files compacted using Dalvik holds the file type name '.dex'. Google states that the credit for Androids successful

development goes to Dalvik VM, because this type of virtual machine, delivers a good performance over various stages of an application runtime environment, conserving more battery-power during long run of an application.

1.4.5 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.

Power Management:

Based on the standard Linux Power Management, Android has its own component. A light-weight power management driver built top of it. CPU shouldn't consume power if no applications or services require power.

BINDER (IPC) DRIVER:

Driver to facilitate inter-process communication between applications and services. The binder driver provides high performance through shared memory, synchronous calls between processes.

1.5 ABOUT PROJECT:

This project works is aimed for developing an efficient food ordering system that can be used in the food & beverage (**F&B**) industry which can help the restaurants to quickly and easily manage daily operational task as well as improve the dining experience of customers. It is believed that still have a lot of restaurants are using the traditional method for food ordering processes. By using the traditional method, it arises a lot of human error while the restaurant's employees deal with large number of customers, this issue will do a great impact to the restaurant in terms of profitability. Thus, this project is to propose a suitable food ordering system for F&B industry to solve the problem that mentioned above. The system will become important tools use for restaurant to improve the management aspect by utilizing computerized system to coordinate each and every food ordering transaction instead of traditional method. In addition, it can also provide efficiency for the restaurant by reducing time consuming, minimize human errors and providing good quality customer service. In terms of the integrity and availability of the system provided, it can be concluded that this system is a suitable solution for the F&B industry.

1.6 PROJECT OBJECTIVES:

Provide convenience for both employees and consumers:

The system will provide an experience of convenience to the restaurant employees while they are on duty as well as the consumer who dine-in at the restaurant. This system allows the staff to serve customers with the minimal delay compare to the paper-based order system, because what the staff need to do is just record down the food that the customer wish to order then the staff place an order via the computer, the food order will be send to the kitchen computer simultaneously. After the order have been successfully placed one copy of the food order with it details will be printed out for customer review. It significantly shortens the time needed to take an order, assume that kitchen area is on ground floor but currently the staff is taking order at second floor. If the restaurant is using paper-based system, the staff has to deliver the food order to ground floor and walk all the way back to second floor, it takes a lot of time and time consuming. Therefore, by using this system it can eliminate this minor section of the order taking process. Besides, it can let consumers to enjoy their meals within a short period of time and thus it can increase the satisfaction and turnover rate of the consumers.

Chapter 2

REQUIREMENT ANALYSIS

2.1 Software Requirements

- Operating System: WINDOWS 7/8/10
- Compiler Used: Android Studio
- Java
- Android SDK (Software Development Kit)

2.2 Hardware Requirements

- Processor Speed: 800 MHz
- RAM Size: 2048 MB DDR
- Keyboard: Standard qwerty serial
- Mouse: Standard serial mouse
- CD-ROM: Speed 48x and above
- Cache memory: 256 KB
- Display: 800*600 or higher resolution display with 256 colors.
- Android device (Version > 2.1)

DESIGN

3.1 Android - UI Controls:

Android provides solid support for the development of UI-based applications. Android provides a variety of widgets that the application programmer can use to create a desired layout and interface. These layout elements can be created via the programming language directly, or through XML layout files

3.2 XML-Based Layouts in Android:

- ❖ XML is a very popular and widely-used format. Hence, a lot of developers are quite comfortable with it.
 - ❖ It helps to provide separation of the UI from the code logic. This provided flexibility to change one without affecting much the other.
 - ❖ Generating XML output is easier than writing direct code, making it easier to have drag-and-drop UI tools to generate interfaces for android apps, i.e.:
-
- Absolute Layout
 - Frame Layout
 - Linear Layout
 - Relative Layout
 - Table Layout
 - Percent Relative Layout
 - Grid Layout
 - Coordinator Layout
 - Constraint Layout
 - Toolbar Layout

Absolute Layout:

In absolute layout, we can specify the exact coordinates of each control that we want to place. In absolute layout, we will give the exact X and Y coordinates of each control. The following is an example of an absolute layout:

Frame Layout:

Frame layout is used when you want to show one item on each screen. Using frame layout, we can have multiple items, but they will be overlapping and only, displaying themselves one at a time. Frame Layout is particularly useful when you want to create animation or movement on screen.

Linear Layout:

Linear layout is used to place one element on each line. So, all the elements will be place in an orderly top-to-bottom fashion. This is a very widely-used layout for creating forms on Android. We are now going to create a small app to display a basic form using the linear layout. The layout.xml file is as follows:

Relative Layout:

Using relative layout, we can specify the position of the elements in relation to other elements, or in relation to the parent container.

Table Layout:

Using table layout, we create a table with rows and columns and place elements within them. In each row, you can specify one or more elements.

Percent Relative Layout:

Percent Relative Layout in Android is a subclass of Relative Layout that supports percentage-based margin and dimensions for Views (Button, Text View or any other view).

Grid Layout:

Android Grid View shows items in two-dimensional scrolling grid (rows & columns) and the grid items are not necessarily predetermined but they automatically inserted to the layout using a List Adapter

Constraint layout:

A Constraint Layout is a View Group which allows you to position and size widgets in a flexible way.

Coordinator Layout:

Coordinator Layout is a super-powered Frame Layout.

Coordinator Layout is intended for two primary use cases:

- As a top-level application decor or chrome layout
- As a container for a specific interaction with one or more child views

Toolbar Layout:

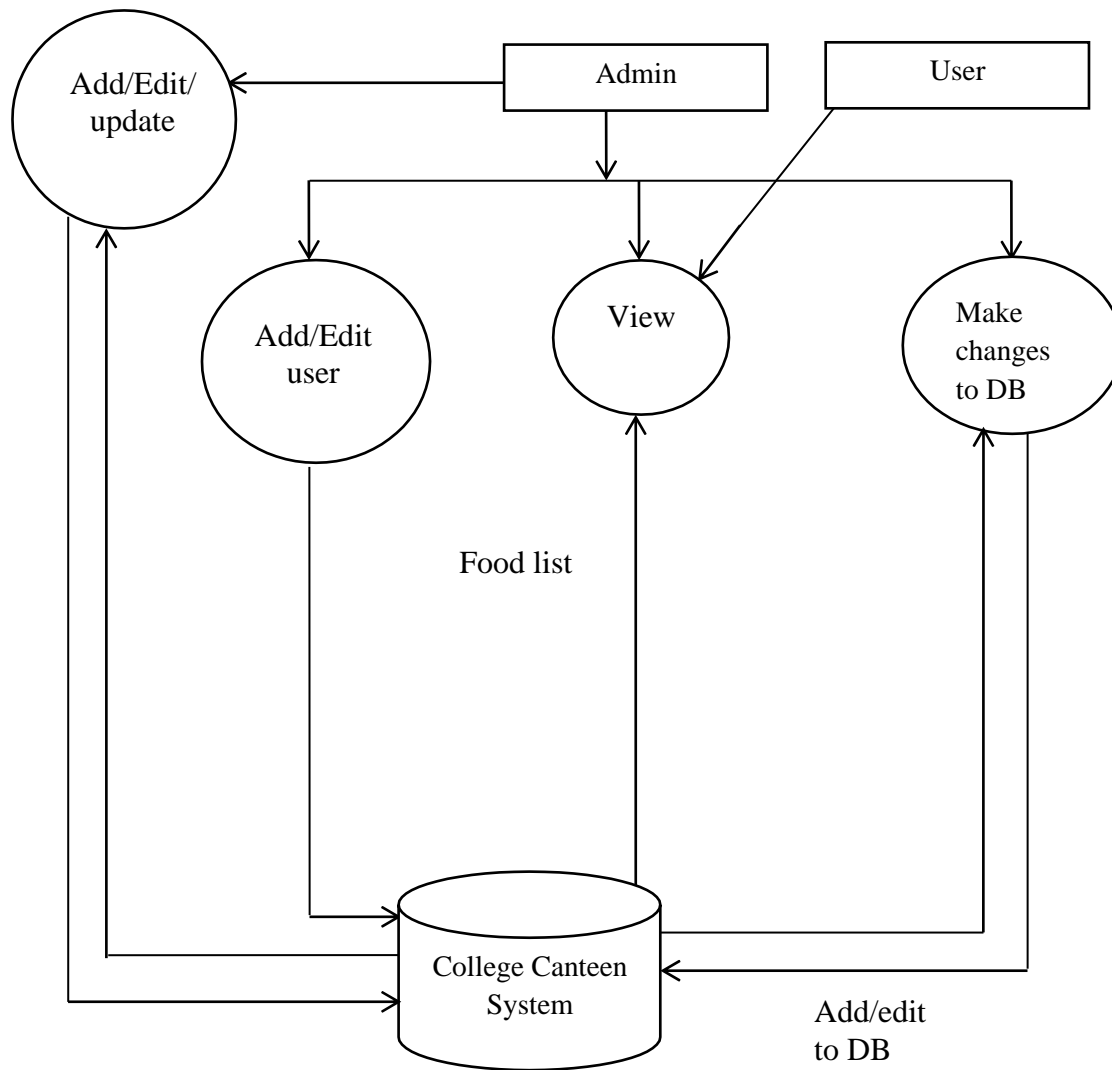
Android Tool Bar can be used as action bar and it can contain navigation button, brand logo, title, subtitle, custom views, and action menu .The difference between action bar and tool bar is that Tool Bar element can be controlled and part of application layout while action bar is controlled by android framework. You can read action bar tutorial for more information on action bar.

In android UI control there are number of UI controls provided by Android that allow you to build the graphical user interface for your app.

Sr. No.	UI Control & Description
1	<u>Text View</u> This control is used to display text to the user.
2	<u>Edit Text</u> Edit Text is a predefined subclass of Text View that includes rich editing capabilities.
3	<u>Auto Complete Text View</u> The Auto Complete Text View is a view that is similar to Edit Text, except that it shows a list of completion suggestions automatically while the user is typing.
4	<u>Button</u> A push-button that can be pressed, or clicked, by the user to perform an action.
5	<u>ImageButton</u> An ImageButton is an Absolute Layout which enables you to specify the exact location of its children. This shows a button with an image (instead of text) that can be pressed or clicked by the user.
6	<u>Check Box</u> An on/off switch that can be toggled by the user. You should use check box when presenting users with a group of selectable options that are not mutually exclusive.
7	<u>Toggle Button</u> An on/off button with a light indicator.
8	<u>Radio Button</u> The Radio Button has two states: either checked or unchecked.
9	<u>Radio Group</u> A Radio Group is used to group together one or more RadioButtons.

10	<u>Progress Bar</u> The Progress Bar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.
11	<u>Spinner</u> A drop-down list that allows users to select one value from a set.
12	<u>Time Picker</u> The Time Picker view enables users to select a time of the day, in either 24-hour mode or AM/PM mode.
13	<u>Date Picker</u> The Date Picker view enables users to select a date of the day.

Page 14



IMPLEMENTATION

4.1 App components

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter your app. Some components depend on others. There are four different types of app components:

- Activities
- Services
- Broadcast receivers
- Content providers

Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. The following sections describe the four types of app components.

Activity:

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. An activity facilitates the following key interactions between system and app:

- Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
- Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
- Helping the app handle having its process killed so the user can return to activities with their previous state restored.
- Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows.

Services:

A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user

interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it.

Broadcast receivers:

A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running. So, for example, an app can schedule an alarm to post a notification to tell the user about an upcoming event... and by delivering that alarm to a `BroadcastReceiver` of the app, there is no need for the app to remain running until the alarm goes off.

Content providers:

A content provider manages a shared set of app data that you can store in the file system, in an SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it. For example, the Android system provides a content provider that manages the user's contact information.

4.3 Functions

protected void onCreate(Bundle savedInstanceState):

The `savedInstanceState` is a reference to a `Bundle` object that is passed into the `onCreate` method of every Android Activity. Activities have the ability, under special circumstances, to restore themselves to a previous state using the data stored in this bundle.

super.onCreate(savedInstanceState) :

By calling `super.onCreate(savedInstanceState);` you tell the Dalvik VM to run your code in addition to the existing code in the `onCreate()` of the parent class. The code in the framework classes handles stuff like UI drawing, house cleaning and maintaining the Activity and application lifecycles.

setContentView setContentView(R.layout.main) :

R means Resource. layout means design. main is the xml you have created under res->layout->main.xml. Whenever you want to change the current look of an Activity or when you move from one Activity to another, the new Activity must have a design to show.

protected void onCreate(Bundle savedInstanceState):

The savedInstanceState is a reference to a Bundle object that is passed into the onCreate method of every Android Activity. Activities have the ability, under special circumstances, to restore themselves to a previous state using the data stored in this bundle.

buttonDone.setOnClickListener(new View.OnClickListener()):

setOnClickListener method is the one that actually listens to the button click. Moving on further, OnClickListener is an Interface definition for a callback to be invoked when a view (button in your case) is clicked.

new Handler().postDelayed(new Runnable()):

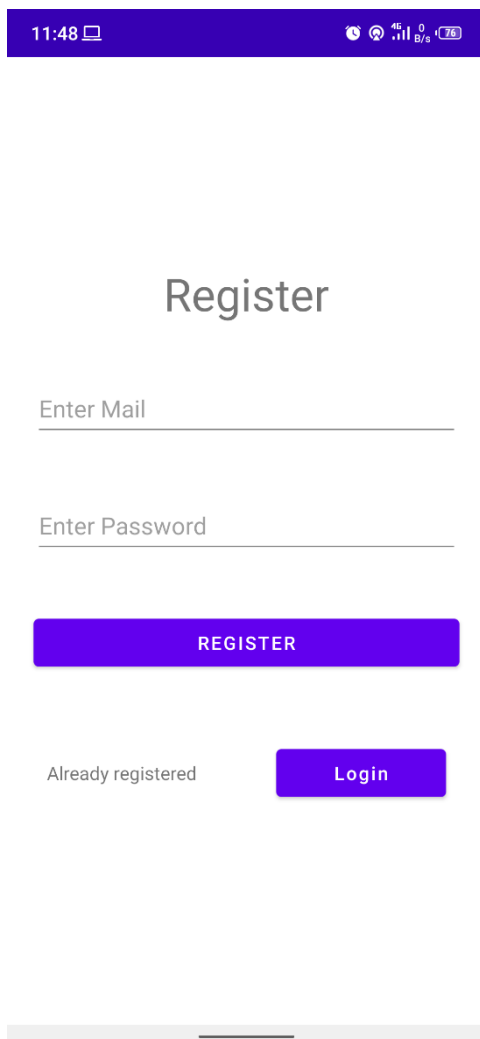
A Handler allows you to send and process Message and Runnable objects associated with a thread's MessageQueue Runnable, long) , postDelayed(Runnable, Object, long) , sendEmptyMessage(int) , sendMessage(Message) , sendMessageAtTime(Message, long) , and sendMessageDelayed(Message, long) methods.

super.onActivityResult(requestCode, resultCode, data):

The android startActivityForResult method, requires a result from the second activity (activity to be invoked). In such case, we need to override the onActivityResult method that is invoked automatically when second activity returns result.

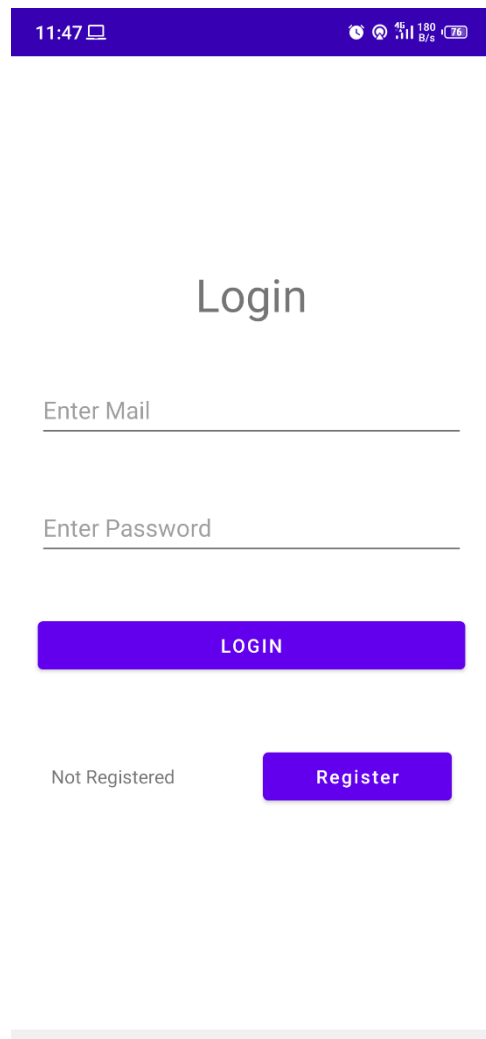
Chapter 5

SNAPSHOTS



The image shows a mobile app interface for the Register page. At the top, there is a status bar with the time 11:48, a laptop icon, and various system icons. Below the status bar, the word "Register" is centered in a large, bold, black font. Underneath, there are two input fields: "Enter Mail" and "Enter Password", each with a horizontal line for text entry. Below the input fields, there is a large, rounded rectangular button with the text "REGISTER" in white. At the bottom, there is a link that says "Already registered" followed by a smaller, rounded rectangular button with the text "Login" in white. The entire interface is set against a light gray background.

Fig 5.1
It is the sign-up page.



The image shows a mobile app interface for the Login page. At the top, there is a status bar with the time 11:47, a laptop icon, and various system icons. Below the status bar, the word "Login" is centered in a large, bold, black font. Underneath, there are two input fields: "Enter Mail" and "Enter Password", each with a horizontal line for text entry. Below the input fields, there is a large, rounded rectangular button with the text "LOGIN" in white. At the bottom, there is a link that says "Not Registered" followed by a smaller, rounded rectangular button with the text "Register" in white. The entire interface is set against a light gray background.

Fig 5.2
It is Login page of app

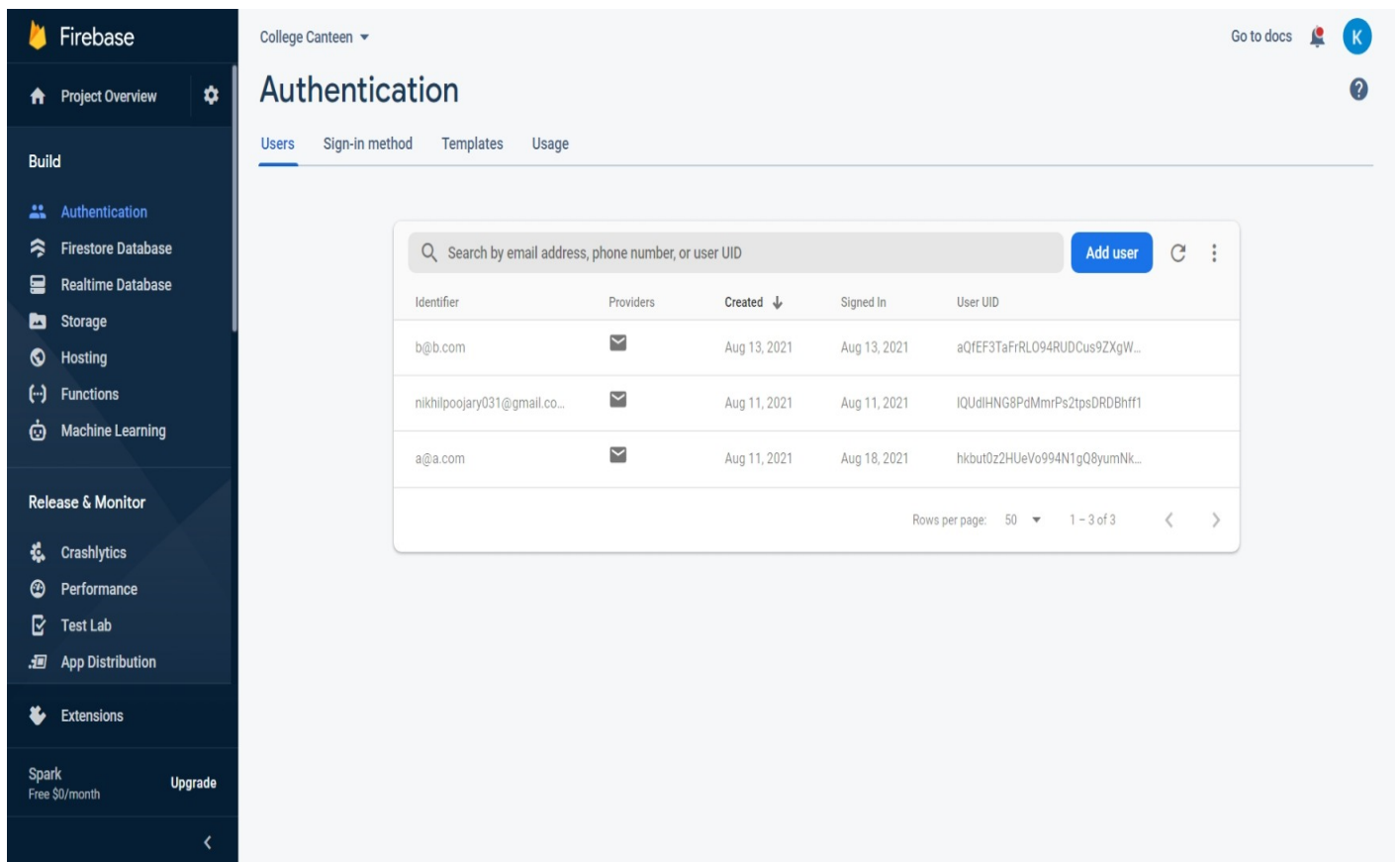


Fig 5.3

This is Firebase Authentication Page on successful register user record will save here

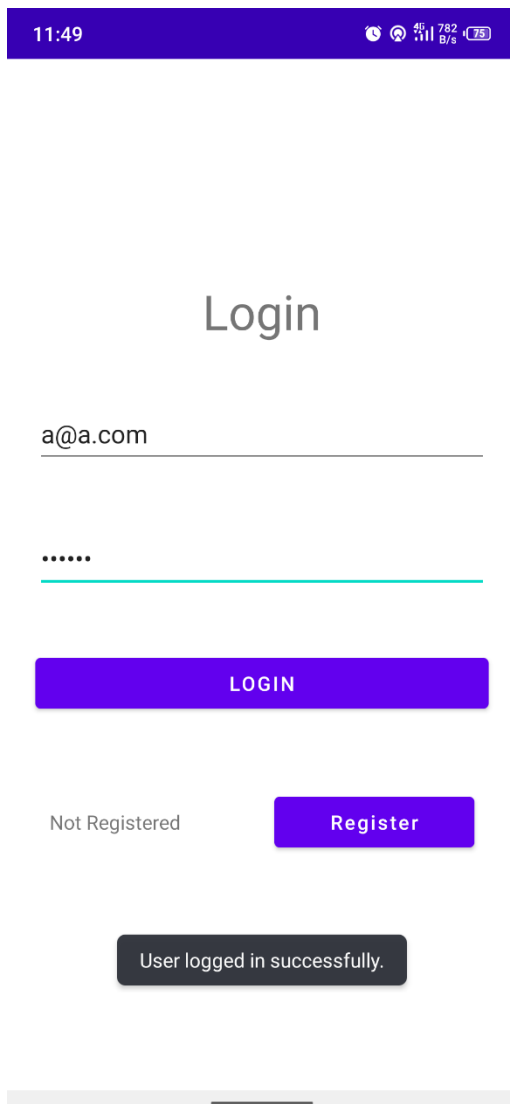
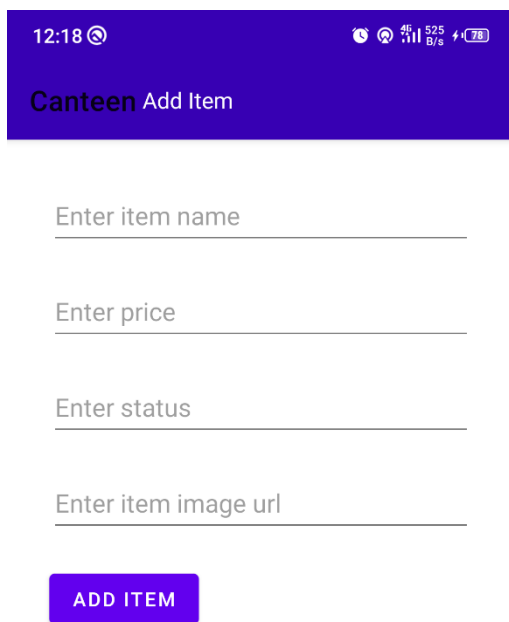


Fig 5.4
Snack bar Message on successful
logged in .

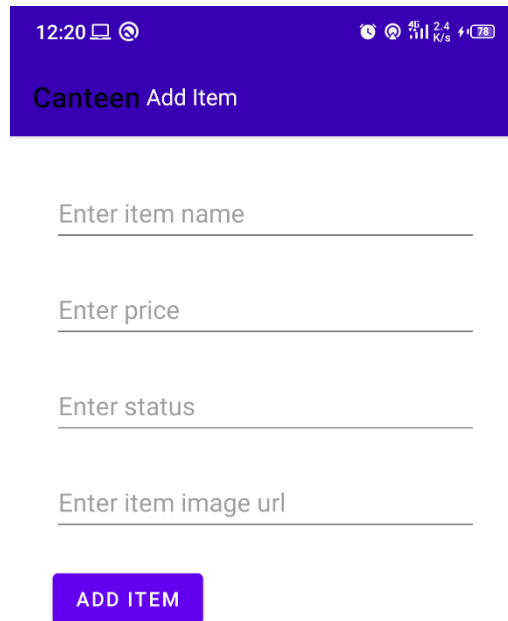


Fig 5.5
This is initial empty admin
page.



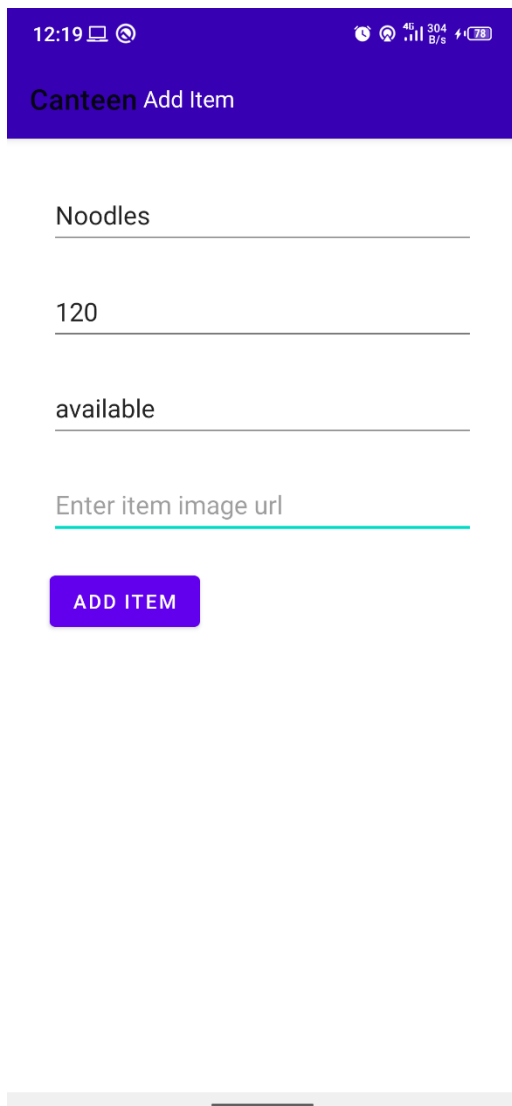
The screenshot shows a mobile application interface with a dark blue header. The header contains the time '12:18', a location icon, and system status icons (signal, 4G, 525 B/s, battery at 78%). Below the header, the text 'Canteen Add Item' is displayed. The main area contains four text input fields with placeholder text: 'Enter item name', 'Enter price', 'Enter status', and 'Enter item image url'. At the bottom, there is a red button labeled 'ADD ITEM'.

Fig 5.6
Here admin can add the required food item.



This screenshot shows the same mobile application interface as Fig 5.6, but with a toast message displayed. The toast message is a dark grey box with rounded corners, containing the text 'Fill all fields.' in white. The input fields and the 'ADD ITEM' button are still visible in the background.

Fig 5.7
Snack bar will appear when admin click on add item button with empty text field.



12:19

Canteen Add Item

Noodles

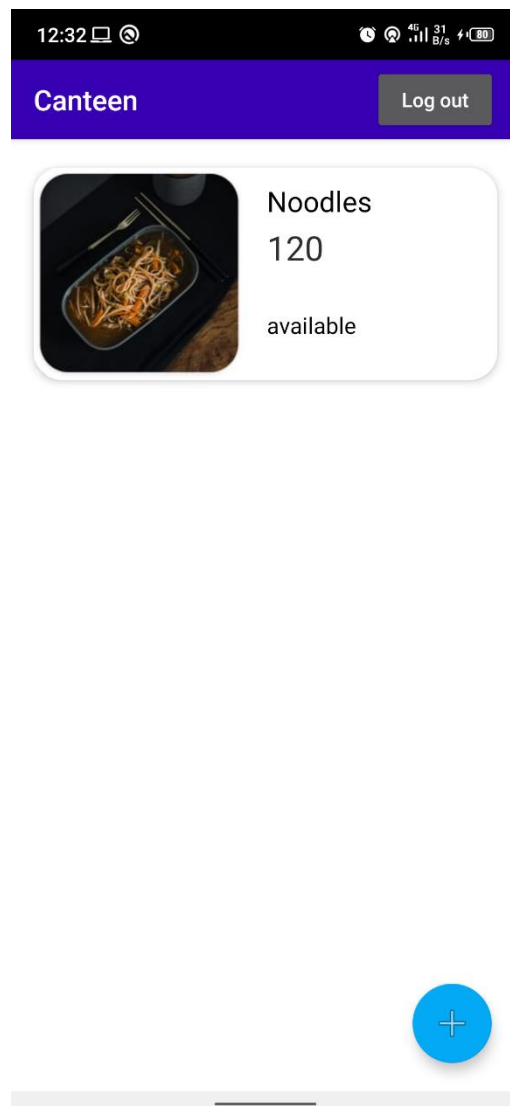
120

available

Enter item image url

ADD ITEM

Fig 5.8
After clicking on add item the
item is saved in firestore.



12:32

Canteen Log out

Noodles
120
available

+

Fig 5.9
This is home page of admin
after adding item.

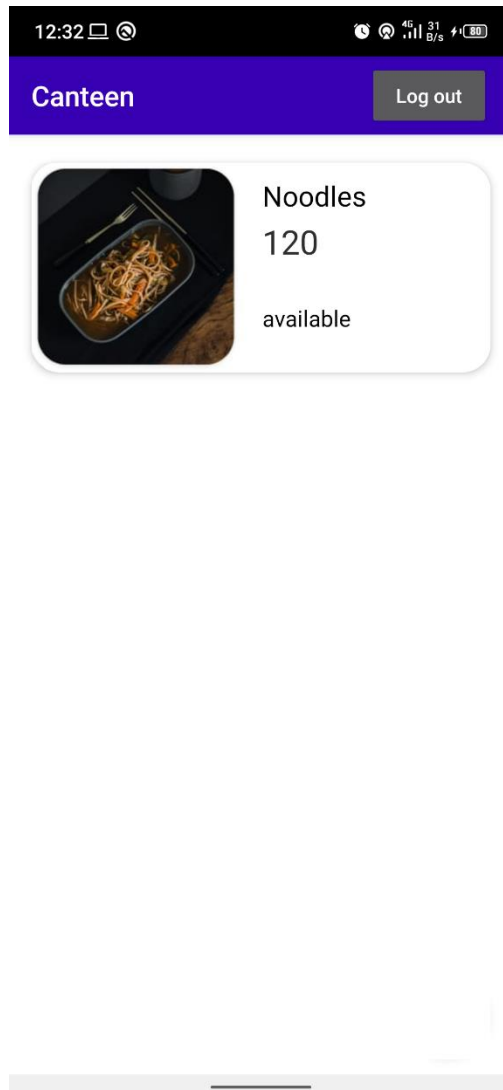


Fig 5.10

This is user home page where user can only see the item

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION:

After a decade, the advancement and innovation of technology help people to manage their task easily and efficiently. In many other industry areas have been used management system to assist their business grow long time ago, therefore it is also a trend that cause F&B industry to make use of a management system for their business. At the end of this project, the system can reduce and replace the human manpower task, reduce the time consume for each transaction and generate report for further management purpose by fully utilizing the system. Obviously, the propose system can help improve the productivity of the restaurant and thus directly did an impact to the profitability of the restaurant. Furthermore, it can also help restaurant to reduce the cost of operation in term of manpower, because the system has already facilitated majority of the business process by using the system. Therefore, it is believed that the system can lead the restaurant's business grow from time to time On the other hand, the technology nowadays allows the portability requirement easy to achieve. Therefore, portability has become one of the factors that have to take into consideration in the system development process. Because portability brings a lot of benefit to user while they using the system such as it provides convenience, accessibility, easy to communicate and etc. Hence, portability has done an impact to the social that everybody is much more preferable to complete their task with portable device. In order to fulfill these all requirement, our proposed method is combined the food ordering system which is in mobile platform into the restaurant management system which is in computer platform. The integration of both features which develop a system that can let user to have an experience of portability which is user can process their food ordering through using their smart phone or tablet. Besides, restaurant manage their daily operation management through using the computer platform it is because computer have some other features such as it has a wider screen, other compatible system that can help to manage the restaurant and some other driver that needed to communicate with those necessary hard wares.

FUTURE ENHANCEMENT:

The system can implement a feature which is real time notification from the mobile phone application to the service desk. This feature enable customer to request customer service through using the mobile application rather than verbally call restaurant staff to approach them. In addition, the mobile application also can implement a feature that allow customer to update the food serve status. For example, customers fine dining at the restaurant they can request the food to be serve through using the mobile application and if the customer finish the main course and feeling full, the customer may request do not serve the following food through using the mobile application. Last but not lease, the mobile application may implement some mini game that is able to entertain customers while they are waiting for the food to be served.

Chapter 7

BIBLIOGRAPHY/REFERENCE

- 1) H.M. Deitel and P.J. Deitel, Java How to program: Sixth Edition
Herbert Schildt, The Complete Reference: Fifth edition
- 2) M. J. P. Wolf, Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming, Greenwood Publishing Group, 2012

User Guide

Run on a real device:

Set up your device as follows:

Connect your device to your development machine with a USB cable. If you developed on Windows, you might need to install the appropriate USB driver for your device.

Perform the following steps to enable **USB debugging** in the **Developer options** window:

Open the Settings app.

If your device uses Android v8.0 or higher, select System. Otherwise, proceed to the next step.

Scroll to the bottom and select About phone.

Scroll to the bottom and tap Build number seven times.

Return to the previous screen, scroll to the bottom, and tap Developer options.

In the Developer options window, scroll down to find and enable USB debugging.

Run the app on your device as follows:

1. In Android Studio, select your app from the run/debug configurations drop-down menu in the toolbar.
2. In the toolbar, select the device that you want to run your app on from the target device drop-down menu.