Ministry of high education,
Culture and science city at Oct 6,
The High institute of computer Science & information systems

المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation Project:

students attendance system

Assistant:

Eng: Amira Gaber

Supervised by

Dr: Ashraf Fahmy

Project No:2106 Academic Year :2021/2022

Ministry of high education,
Culture and science city at Oct
6,
The High institute of computer Science & information systems

المعهد العالي لعلوم الحاسب ونظم المعلومات

Graduation Project:

Students attendance system

Prepared by

محمد علي ابراهيم ١٢٣٤٥٦ احمد صديق سلامه ١٢٣٤٥٦

احمد محمود سيد ١٢٣٤٥٦ ايهاب احمد محمد ١٢٣٤٥٦

عبدالرحمن رفعت حسن ١٢٣٤٥٦

Assistan

Eng: Amira Gaber

Supervised by

Prof.Dr: **Dr:** Ashraf Fahmy

Project No: 2106…… Academic Year :2021/2022

Acknowledgement

I express my gratitude to. Dr: Ashraf Fahmy

And Educational Institution The High institute of computer Science & information systems for having provided me the facilities to do the project successfully.

My heartful thanks to Dr. Ashraf Fahmy giving us an opportunity to undertake this Project.

My sincere thanks to Prof Dr.Sami Abd El Moneim Al-Dalil who has allowed me to do this project and encouragement given to me. I owe deep sense of gratitude to for appreciating my goal.

Eng:Amira Gaber I express my sincere thanks to her for her constant encouragement.

I would also like to thank my Project Co-ordinator Dr: Ashraf Fahmy for his valuable guidance and support to meet the successful completion of my project.

I express my sincere thanks to staff encouragement and valuable guidance throughout this project.

Last but not the least; I extend my sincere thanks to my family members and my friends for their constant support throughout this project

Project Abstract

Student attendance management system deals with the maintenance of the student's attendance details. It is generates the attendance of the student on basis of presence in class. It is maintained on the daily basis of their attendance. the staffs will be provided with the separate username & password to make the student's status.

The staffs handling the particular subjects responsible to make the attendance for all students. Only if the student present on that particular period, the attendance will be calculated. The students attendance reports based on weekly and consolidate will be generated.

Table of Content

Acknowledgm..... i

Abstract.....ii

Table of Content 5

1 Chapter One Introduction 9

1.1 Purpose ……..10

1.2 Scope…….…10

1.3 Definitions, acronyms and abbreviations……..11

1.4 General Description……...……11

1.5 Application Function ……11

1.6 User Characteristics…….……..12

1.7 Assumptions and Dependencies…….……12

1.8 Functional Requirements……13

1.9 Non- Functional Requirements……13

1.1.1 Hardware Specification……..……...14

1.1.2 Software Specification……..……14

1.1.3 Problem Definition……..……15

1.1.4 Project Overview ……..…15

1.2 Project Tools……..……..……16

1.2.1 Adobe Photoshop ……..……..16

1.2.2 Adobe XD……...16

1.2.3 Android Studio……...…….……17

1.2.4 Emulator Blue Stacks 5…….……..18

1.2.5 Fir Data Base……...……..…….……...19

2 Chapter Two System Analysis …….21

2.1 Proposed System……..22

2.2 Diagram design…….…….…..23

2.3 System Maintenance……..…….……24

3 Chapter Three: System Design and Implementation……...…..….. 26

3.1 Splash Screen ……..……..26

3.2 Login Screen ……..27

3.3 Register Screen……28

3.4 QR Code Screen ……....29

3.5 Splash Screen Source Code ……...30

3.6 Login Screen Source Code …….....33

3.7 Login Screen Source Code ……34

3.8 Register Screen Source Code …….36

3.9 QR Code Screen Source Code …….……41

4 Chapter Four : Conclusion and future work.....45

References ……...…..46

List of Figures

Figure 1.1 Adobe Photoshop logo ……..16

Figure 1.2 Adobe XD……...…16

Figure 1.3 Android Studio logo ……...17

Figure 1.4 Emulator Blue Stacks 5 logo……...18

Figure 2.1 fire base……...19

Figure 2.2 ERD Diagram……..23

Figure 3.1 Splash Screen……..26

Figure 3.2 Login Screen……27

Figure 3.3 Register Screen……28

Figure 3.4 QR Code Screen……..29

Figure 4.1 Splash Screen Source Code ……..…30

Figure 4.2 Login Screen Source Code ……...33

Figure 4.3 Register Screen Source Code ……..34

Figure 4.4 QR Code Screen Source Code ……36

Chapter one

CHAPTER 1

INTRODUCTION

“Attendance Management System” is software developed for maintaining the attendance of the student on the daily basis in the collage.

Here the staffs, who are handling the subjects, will be responsible to mark the attendance of the students.

Each staff will be given with a separate username and password based on the subject they handle.

An accurate report based on the student attendance is generated here.

This system will also help in evaluating attendance eligibility criteria of a student. Report of the student’s attendance on weekly and monthly basis is generated.

1.1 Purpose :

The purpose of this document is to provide a detailed description of the Student Attendance Management System. It will explain the purpose and features of the program, the interfaces of the program, what the program will do, the constraints under which it must operate and how the program will react to external stimuli. This document is intended for both users and developers of the software

1.2 Scope :

This document covers the requirements for the Student Attendance Management System.
This software will provide a graphical environment in which the users of the system will be able to perform various operations that are associated with storing, marinating, updating and retrieving Student’s Attendance information. The system will capture information about Student's and Professor's
personal details and courses. Storing, updating and retrieving in a fast and accurate way.

1.3 Definitions, acronyms and abbreviations:

The Student Attendance Management System has to handle records for many number of students and maintenance was difficult. Though it has used an information system, it was totally manual.
Hence there is a need to upgrade the system with a computer based information system. Student Management System

1.4 General Description:

Application Perspective

The application Student Attendance Management system is an independent product and does not depend on any other product or system. The product will automate various tasks associated with handling student details and better organizing the stored information and optimum performance, thus helping the Colleges to ensure smooth working of these processes

1.5 Application Functions

Our system has two types of accessing modes,
i) Administrator: Administrator has to update and monitor the registered student details, add a new student, provide register number for all students, assign each student a course etc., Administrator can Also add the professor's de
Tails and create a separate login for him/her and assign that particular course handled by them, and also can give help to the Professors in usage of Student Attendance Management System

ii) User: There are only one users login created Professor:
Professor can get logged in, mark attendance, checkup the results of the student updated by admin and can also inform the students about their shortage in that particular course

1.6 User Characteristics:

This software gives access to two kinds of users.
1. Administrator: The personnel and College administrator will have administrator access to add, delete and modify information stored in the database
2. Authorized User: Professors will have access to only view the data stored in the database
And can update the student's attendance in the form of formatted reports

1.7 Assumptions and Dependencies:

We assume that the Office personnel do all the data entry based and the correct values obtained from forms and registers.

We assume that the computers that will use the software will be part of the college.

Users with administrator access should be careful in deleting or modifying any information knowingly or unknowingly which will lead to inconsistency of the database.

1.8 Functional Requirements:

- Student Attendance Management System involves the following functions

Student Registration:

- Details of students is maintained and entered at the back end by Administrator

Student Attendance Management:

- Easily track attendance information of students.
- Quickly alerts the shortage of the attendance of particular subjects.

1.9 NON-FUNCTIONAL REQUIREMENTS:

The Non-Functional requirements of our project are:

1. **Time :**

   This project should be completed within the stimulated time period.

2. **Cost :**

   The cost involved in marketing the project should be less.

3. **Usability:**

   This requirement is present, as this system will interact with the user.

4. **Reliability:**

   This system must be highly robust

5. **Performance:**

   It would be fast enough to produce output.

SYSTEM SPECIFICATION

3.1 Hardware Reqirements (Minimum Requirement)

- Minimum RAM: 4 GB
- Hard Disk: 256 GB
- Processor:-Intel Pentium 4(2.20 GHZ) or above

3.2 Software Requirements (minimum Requirement)

- Operating system :Windows 8.1
- Ux. Ui : Photoshop . Adobe XD
- Front-End Language: xml Language
- Back-End Language: kotlin Language
- Back-End Connectivity: android studio
- Back-End : fire data base
- Emulator
- Diagram design
- Fir data base

Problem Definition :

This system developed will reduce the manual work and avoid redundant data. By maintaining the attendance manually, then efficient reports cannot be generated. The system can generate efficient weekly ,consolidate report based on the attendance. As the attendances are maintained in registers it has been a tough task for admin and staff to maintain for long time. Instead the software can keep long and retrieve the information when needed.

Project Overview :

Attendance Management System basically has two main modules for proper functioning

- Admin module is has rights for creating any new entry of faculty and student details.
- User has a rights of making daily attendance, generating report. Attendance report can be taken by given details of student details, date, class

Project Tools

Development Tools :

1.2.1) Adobe Photoshop : is software that is extensively used for raster image editing, graphic design and digital art. It makes use of layering to allow for depth and flexibility in the design and editing process, as well as providing powerful editing

Figure 1.1

tools, that when combined, are capable of just about anything

1.2.2) Adobe XD :

(also known as Adobe Experience Design) is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for macOS and Windows,

Figure 1.2

and there are versions for iOS and Android to help preview the result of work directly on mobile devices

Adobe XD enables website wireframing and creating click-through prototypes.

1.2.3) Android Studio :

is the official integrated development environment (IDE) for Android application development.

It is based on the IntelliJ IDEA,

a Java integrated development environment for software, and incorporates its code editing and developer tools.

Figure 1.3

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

1.2.4) Blue Stacks :

is a popular Android emulator for Windows and Mac. Using BlueStacks, you can run virtually any Android app on your desktop.

Like any emulator,

BlueStacks creates a virtual version of an Android device that runs in a window on your computer.

Figure 1.3

It doesn't look exactly like an Android device, but it resembles a phone's screen well enough that even a first-time user should have no trouble using it.

BlueStacks is free to download, install, and use. While you can use BlueStacks to run almost any Android app (it's compatible with about 97% of the apps in the Google Play Store), the app has found its largest audience with Android users who want to play mobile games on their desktop computer

1.2.5) Fire Data Base :

Figure 2.2

Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company that was founded in 2011. In 2014, the platform was acquired by Google and it is now the premiere of app development.

CHAPER TWO

CHAPTER 2

SYSTEM ANALYSIS

INTRODUCTION

Analysis can be defined as breaking up of any whole so as to find out their nature, function etc.

It defines design as to make preliminary sketches of; to sketch a pattern or outline for plan. To plan and carry out especially by artistic arrangement or in a skillful wall.

System analysis and design can be characterized as a set of techniques and processes, a community of interests, a culture and an intellectual orientation.

The various tasks in the system analysis include the following.

- Understanding application.
- Planning.
- Scheduling.
- Developing candidate solution.
- Performing trade studies.
- Performing cost benefit analysis.
- Recommending alternative solutions.
- Selling of the system.
- Supervising, installing and maintaining the system.

This system manages to the analysis of the report creation and develops manual entry of the student attendance. First design the students entry form , staff allocation and time table allocation forms. This project will helps the attendance system for the department calculate percentage and reports for eligibility criteria of examination .The application attendance entry system will provide flexible report for all students.

The Existing system is a manual entry for the students. Here the attendance will be carried out in the hand written registers. It will be a tedious job to maintain the record for the user. The human effort is more here. The retrieval of the information is not as easy as the records are maintained in the hand written registers. This application requires correct feed on input into the respective field. Suppose the wrong inputs are entered, the application resist to work. so the user find it difficult to use

2.1) PROPOSED SYSTEM

To overcome the drawbacks of the existing system, the proposed system has been evolved. This project aims to reduce the paper work and saving time to generate accurate results from the student's attendance. The system provides with the best user interface. The efficient reports can be generated by using this proposed system.

Advantages of Proposed System

- It is trouble-free to use.
- It is a relatively fast approach to enter attendance.
- Is highly reliable, approximate result from user.
- Best user Interface.
- Efficient reports.

2.2) ERD Diagram :

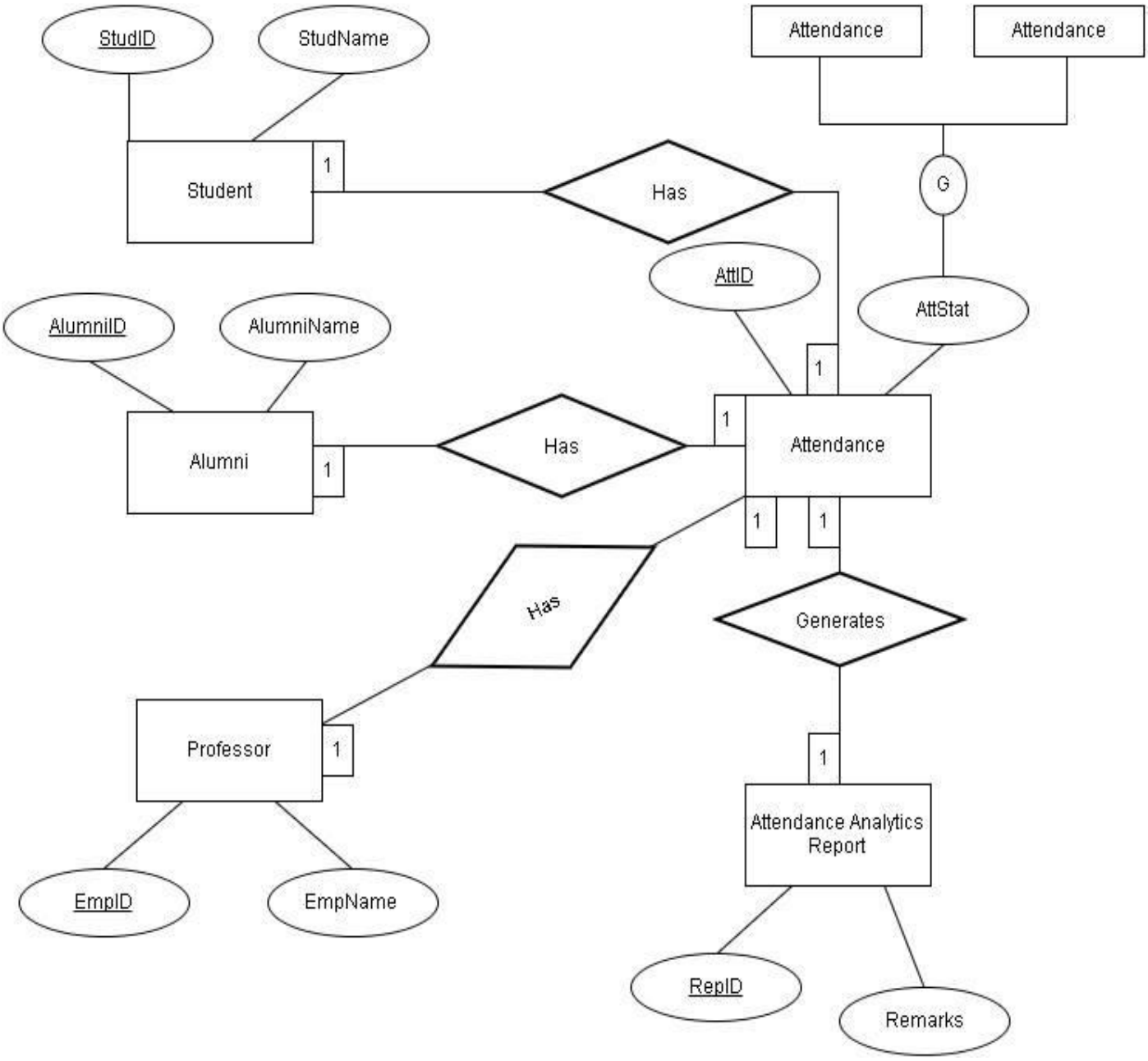

Figure 2.1

2.3) System Maintenance:

Software maintenance is far more than finding mistakes. Provision must be made for environment changes, which may affect either the computer, or other parts of the computer based systems. Such activity is normally called maintenance. It includes both the improvement of the system functions and the corrections of faults, which arise during the operation of a new system. It may involve the continuing involvement of a large proportion of computer department recourses. The main task may be to adapt existing systems in a changing environment. Back up for the entire database files are taken and stored in storage devices like flash drives, pen drives and disks so that it is possible to restore the system at the earliest. If there is a breakdown or collapse, then the system gives provision to restore database files. Storing data in a separate secondary device leads to an effective and efficient maintains of the system. The nominated person has sufficient knowledge of the organization's computer passed based system to be able to judge the relevance of each proposed change.

CHAPTER THREE

CHAPTER 3

System Design and Implementation

Splash Screen :

It is the first screen that appears when opening the application and remains stable for 3 seconds (1500 milliseconds) and then disappears and the application goes to the login page

Figure 3.1

Login Screen :

On this screen, the institute’s logo appears and a field for writing the e-mail and the password, if the user has previously registered. If you have not yet been registered, you can click on register a new user to go to the registration screen

Figure 3.2

Register Screen :

Here you can register as a new user by writing your name, e-mail and your password, taking into account that it is written in a correct way, then pressing the word “Register” and then you can log in (and you can return to the registration page again by clicking on the “I already have an account” button)

Figure 3.3

QR Code Screen :

The last screen for recording the attendance of the lecture through a QR code.

The phone camera is directed to the QR code of the lecture in order to record the attendance

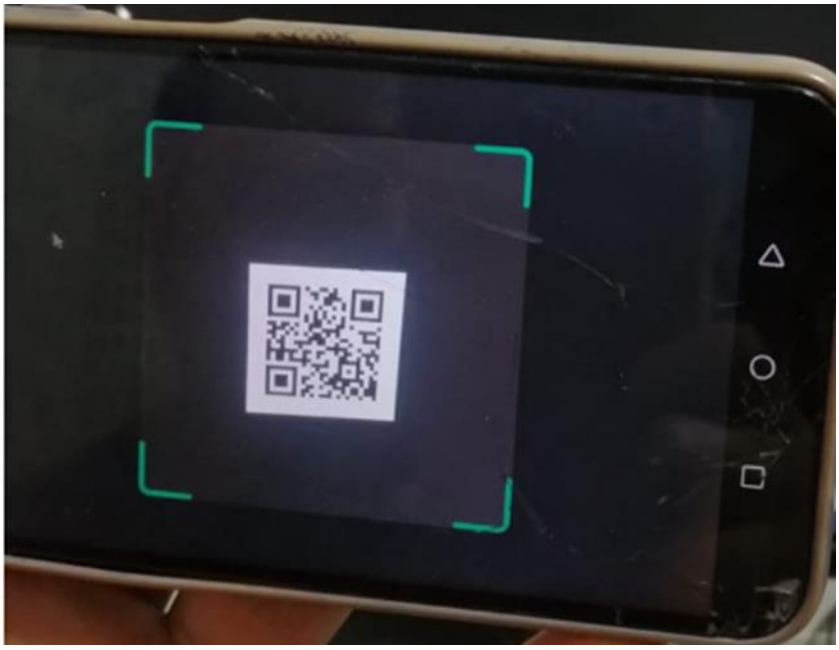

Figure 3.4

Source Code

Splash Screen :

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/black"
    tools:context=".ui.SplashActivity">



    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_logo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:tint="@color/white" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 4.0

Data Base :

```kotlin
package com.kagroup.tailor.data.local

import ...

@Dao
interface UserDAO {
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUserToDatabase(user: User): Long

    @Delete
    suspend fun deleteUser(user: User)

    @Query( value: "SELECT * FROM users_table")
    fun observeAllUsers(): LiveData<List<User>>

    @Query( value: "SELECT  * FROM users_table WHERE mobileNumber = :userMobile AND password = :password")
    fun observeUserByMobileNumber(userMobile: String, password: String): LiveData<User>


    @Query( value: "SELECT  * FROM users_table WHERE mobileNumber = :userMobile")
    fun checkIfUserExists(userMobile: String) : LiveData<User?>
}
```

Figure 4.1

```kotlin
package com.kagroup.tailor.data.local

import ...

@Database(
    entities = [User::class],
    version = 1
)
abstract class UserDataBase : RoomDatabase() {
    abstract fun userDao(): UserDAO
}
```

Figure 4.2

```
package com.kagroup.tailor.data.local

import ...

object UserDataSource {
    fun saveUser(user: User?) {
        preferencesGateway.save(Constants.USER, Gson().toJson(user))
    }

    fun getUser(): User? {
        return Gson().fromJson(
            preferencesGateway.load(USER,  defaultValue: ""),
            User::class.java
        )
    }

    fun hasUser(): Boolean {
        val user = Gson().fromJson(
            preferencesGateway.load(USER,  defaultValue: ""),
            User::class.java
        )

        if (user != null && user.mobileNumber != null &&
            user.mobileNumber.toString().isNotEmpty()) {
            return user.mobileNumber != null
        }
        return false
    }
}
```

Figure 4.3

Login activity kotlin code :

```kotlin
class LoginActivity : BaseActivity<ActivityLoginBinding, LoginViewModel>(), LoginNavigator {

    private val mViewModel: LoginViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun getViewModel(): LoginViewModel {
        return mViewModel
    }

    override fun getLayoutId(): Int {
        return R.layout.activity_login
    }

    override fun init() {
        mViewModel.navigator = this
        viewDataBinding?.vm = mViewModel
        viewDataBinding?.navigator = this
    }

    override fun subscribeToLiveData() {
        mViewModel.userIsExistsLiveData.observe( owner: this) { mEvent ->
            mEvent.getContentIfNotHandled()?.let { user ->
                if (user.id != null) {
                    UserDataSource.saveUser(user)
                    openMain()  ^let
                } else {
                    showPopUp("خطا في البيانات", "موافق",  isCancelable: false)  ^let
                }
            } ?: run {  this: LoginActivity
                showPopUp("خطا في البيانات", "موافق",  isCancelable: false)
            }
        }
```

Figure 4.4

```kotlin
    override fun openMain() {
        val intent = Intent( packageContext: this, MainActivity::class.java)
        intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
        startActivity(intent)
    }

    override fun openRegister() {
        startActivity(Intent( packageContext: this, RegisterActivity::class.java))
    }
}
```

kotlin code

Login view model:

```
class LoginViewModel @ViewModelInject constructor(
    private val userRepository: UserRepository
): BaseViewModel<LoginNavigator>() {
    com.base BaseViewModel.kt
    public abstract class BaseViewModel<N> : ViewModel
    Tailor.app

    var mobileError = ObservableBoolean( value: false)
    var passwordError = ObservableBoolean( value: false)


    val userIsExistsLiveData = MediatorLiveData<Event<User?>>()


    init {
        mobileText.addCallback { it: String?
            if (it.toString().isNotEmpty()) {
                mobileError.set(false)
            }
        }
        passwordText.addCallback { it: String?
            if (it.toString().isNotEmpty()) {
                passwordError.set(false)
            }
        }
    }
```

Figure 4.6

```
    private fun isValidLogin(): Boolean {
        var isValid = true

        if (!ValidationUtils.isValidMobile(mobileText.get().toString())) {
            isValid = false
            mobileError.set(true)
        }
        if (!ValidationUtils.isValidPassword(passwordText.get().toString())) {
            isValid = false
            passwordError.set(true)
        }
        return isValid
    }

    val loginClick = View.OnClickListener { it: View!
        if (isValidLogin()) {
            login()
        }
    }

    private fun login() {
        setShowLoadingLayout(true)
        viewModelScope.launch(Dispatchers.IO) { this: CoroutineScope
            val result = userRepository
                .observeUserByMobileNumber(mobileText.get().toString(),
                    passwordText.get().toString())
            delay( timeMillis: 1000)
            setShowLoadingLayout(false)
            withContext(Main) { this: CoroutineScope
                userIsExistsLiveData.addSource(result) { user ->
                    userIsExistsLiveData.postValue(Event(user))
                }
            }
        }
    }
}
```

Register activity :

```
class RegisterActivity : BaseActivity<ActivityRegisterBinding, RegisterViewModel>(),
    RegisterNavigator {

    private val mViewModel: RegisterViewModel by viewModels()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun getViewModel(): RegisterViewModel {
        return mViewModel
    }

    override fun getLayoutId(): Int {
        return R.layout.activity_register
    }

    override fun init() {
        mViewModel.navigator = this
        viewDataBinding?.vm = mViewModel
        viewDataBinding?.navigaor = this
    }

    override fun subscribeToLiveData() {
        mViewModel.userIsExistsLiveData.observe( owner: this) { mEvent ->
            mEvent.getContentIfNotHandled()?.let { user ->
                if (user.id != null) {
                    showPopUp("هذا المستخدم موجود بالفعل", "موافق", isCancelable: false) ^let
                } else {
                    mViewModel.register() ^let
                }
            } ?: run { this: RegisterActivity
                mViewModel.register()
            }
        }
    }
```

Figure 4.8

```
override fun openMain() {
    val intent = Intent( packageContext: this, MainActivity::class.java)
    intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
    startActivity(intent)
}

override fun backToLogin() {
    super.onBackPressed()
}
}
```

Figure 4.9

Register view model :

```
class RegisterViewModel @ViewModelInject constructor(
    private val userRepository: UserRepository
) : BaseViewModel<RegisterNavigator>() {

    var passwordText = ObservableField( value: "")
    var mobileText = ObservableField( value: "")
    var nameText = ObservableField( value: "")

    var passwordError = ObservableBoolean( value: false)
    var mobileError = ObservableBoolean( value: false)
    var nameError = ObservableBoolean( value: false)

    val userIsExistsLiveData = MediatorLiveData<Event<User?>>()

    private lateinit var mUser: User


    init {
        passwordText.addCallback { it: String?
            if (it.toString().isNotEmpty()) {
                passwordError.set(false)
            }
        }
        mobileText.addCallback { it: String?
            if (it.toString().isNotEmpty()) {
                mobileError.set(false)
            }
        }
        nameText.addCallback { it: String?
            if (it.toString().isNotEmpty()) {
                nameError.set(false)
            }
        }
    }
```

Figure 4.10

```
private fun isValidRegister(): Boolean {
    var isValid = true

    if (!ValidationUtils.isValidPassword(passwordText.get().toString())) {
        isValid = false
        passwordError.set(true)
    }
    if (!ValidationUtils.isValidMobile(mobileText.get().toString())) {
        isValid = false
        mobileError.set(true)
    }
    if (!ValidationUtils.isValidText(nameText.get().toString())) {
        isValid = false
        nameError.set(true)
    }
    return isValid
}

val registerClick = View.OnClickListener { it: View!
    if (isValidRegister()) {
        mUser = prepareRegisterRequest()
        checkUserIsNotInDB()
    }
}
```

Figure 4.11

Xml Code

Login:

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layoutDirection="rtl"
    tools:context=".ui.login.LoginActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="50dp"
            android:orientation="vertical"
            android:paddingBottom="50dp">

            <ImageView
                android:layout_width="70dp"
                android:layout_gravity="center"
                android:layout_height="70dp"
                android:src="@drawable/ic_logo"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                style="@style/bold_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:paddingStart="30dp"
                android:paddingTop="30dp"
                android:paddingEnd="5dp"
                android:paddingBottom="30dp"
                android:text="تسجيل الدخول"
                android:textColor="@android:color/black"
                android:textSize="20sp" />
```

Figure 5.1

```
<LinearLayout                                                    ⚠ 3  ✓ 2
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="30dp"
    android:layout_marginEnd="30dp"
    android:orientation="vertical">

    <TextView
        style="@style/bold_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="رقم الهاتف"
        android:textColor="@color/colorAccent"
        android:textSize="14sp"
        app:layout_constraintBottom_toBottomOf="@+id/back_imageView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/back_imageView" />

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/edit_text_bg">

        <ImageView
            android:id="@+id/flag"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:background="@color/colorGrey40"
            android:padding="12dp"
            android:src="@drawable/ic_egypt"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
```

Figure 5.2

```
            <EditText
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:background="@android:color/transparent"
                android:digits="0123456789"
                android:hint="01XX XXX XXXX"
                android:inputType="number"
                android:maxLength="11"
                android:text="@={vm.mobileText}"
                android:paddingStart="15dp"
                android:paddingEnd="15dp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toEndOf="@+id/flag" />
        </androidx.constraintlayout.widget.ConstraintLayout>

        <TextView
            style="@style/regular_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:text="برجاء ادخال رقم هاتف صحيح"
            android:textColor="@color/colorError"
            android:textSize="12sp"
            android:visibility="@{vm.mobileError ? View.VISIBLE : View.GONE}"

    </LinearLayout>
```

Figure 5.3

```
            <EditText
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:background="@android:color/transparent"
                android:digits="0123456789"
                android:hint="01XX XXX XXXX"
                android:inputType="number"
                android:maxLength="11"
                android:text="@={vm.mobileText}"
                android:paddingStart="15dp"
                android:paddingEnd="15dp"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toEndOf="@+id/flag" />
        </androidx.constraintlayout.widget.ConstraintLayout>

        <TextView
            style="@style/regular_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:text="برجاء ادخال رقم هاتف صحيح"
            android:textColor="@color/colorError"
            android:textSize="12sp"
            android:visibility="@{vm.mobileError ? View.VISIBLE : View.GONE}"

    </LinearLayout>
```

Figure 5.4

Register :

```
android:layout_marginEnd="30dp"
android:orientation="vertical">

<TextView
    style="@style/bold_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="الاسم بالكامل"
    android:textColor="@color/colorAccent"
    android:textSize="14sp" />

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="15dp"
    android:background="@drawable/edit_text_bg"
    app:hintAnimationEnabled="false"
    app:passwordToggleTint="@android:color/black">

    <EditText
        android:id="@+id/merchant_name_editText"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:background="@android:color/transparen
        android:hint="الاسم بالكامل"
        android:inputType="text"
        android:paddingStart="15dp"
        android:paddingEnd="15dp"
        android:text="@={vm.nameText}"
        android:textColor="@color/colorAccent"
        android:textColorHint="@color/colorBlack30"
        android:textSize="14sp"
        app:layout_constraintBottom_toBottomOf="paren
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 5.5

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:orientation="vertical"
    android:paddingBottom="50dp">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:src="@drawable/ic_logo"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        style="@style/bold_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingStart="30dp"
        android:paddingTop="30dp"
        android:paddingEnd="5dp"
        android:paddingBottom="30dp"
        android:text="تسجيل حساب جديد"
        android:textColor="@android:color/black"
        android:textSize="20sp" />


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginEnd="30dp"
        android:orientation="vertical">
```

Figure 5.6

```
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="30dp"
                android:layout_marginEnd="30dp"
                android:orientation="vertical">

                <TextView
                    style="@style/bold_text"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="20dp"
                    android:text="رقم الهاتف"
                    android:textColor="@color/colorAccent"
                    android:textSize="14sp"
                    app:layout_constraintBottom_toBottomOf="@+id/back_imageView"
                    app:layout_constraintEnd_toEndOf="parent"
                    app:layout_constraintStart_toStartOf="parent"
                    app:layout_constraintTop_toTopOf="@+id/back_imageView" />

                <androidx.constraintlayout.widget.ConstraintLayout
                    android:layout_width="match_parent"
                    android:layout_height="50dp"
                    android:layout_marginTop="10dp"
                    android:background="@drawable/edit_text_bg">

                    <ImageView
                        android:id="@+id/flag"
                        android:layout_width="wrap_content"
                        android:layout_height="match_parent"
                        android:background="@color/colorGrey40"
                        android:padding="12dp"
                        android:src="@drawable/ic_egypt"
                        app:layout_constraintBottom_toBottomOf="parent"
                        app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintTop_toTopOf="parent" />
```

Figure 5.7

```
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center"
                    android:layout_marginTop="20dp"
                    android:layout_marginBottom="20dp"
                    android:text="بالفعل لدي حساب"
                    android:onClick="@{()->navigaor.backToLogin()}"
                    android:textColor="@android:color/black"
                    android:textSize="14sp" />
            </LinearLayout>
        </ScrollView>

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:visibility="@{vm.showLoadingLayout ? View.VISIBLE : View.GONE}"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:visibility="gone">

            <include layout="@layout/loading_layout" />

        </LinearLayout>

    </androidx.constraintlayout.widget.ConstraintLayout>

/layout>
```

Figure 5.8

Main Activity:

```
package com.example.barcodescanner

import android.annotation.SuppressLint
import android.content.pm.PackageManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.SurfaceHolder
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.Toast
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.example.barcodescanner.databinding.ActivityMainBinding
import java.io.IOException
import com.google.android.gms.vision.CameraSource
import com.google.android.gms.vision.Detector
import com.google.android.gms.vision.barcode.Barcode
import com.google.android.gms.vision.barcode.BarcodeDetector
import com.google.android.gms.vision.Detector.Detections
```

Figure 7.1

```
class MainActivity : AppCompatActivity() {
        private val requestCodeCameraPermission = 1001
        private lateinit var cameraSource: CameraSource
        private lateinit var barcodeDetector: BarcodeDetector
        private var scannedValue = ""
    private lateinit var binding: ActivityMainBinding

        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
           binding = ActivityMainBinding.inflate(layoutInflater)
            val view = binding.root
            setContentView(view)


            if (ContextCompat.checkSelfPermission(
                    this@MainActivity,
android.Manifest.permission.CAMERA
                ) != PackageManager.PERMISSION_GRANTED
            ) {
                askForCameraPermission()
            } else {
                setupControls()
            }

            val aniSlide: Animation =
                AnimationUtils.loadAnimation(this@MainActivity,
R.anim.scanner_animation)
            binding.barcodeLine.startAnimation(aniSlide)
        }
```

```
        private fun setupControls() {
            barcodeDetector =

BarcodeDetector.Builder(this).setBarcodeFormats(Barcode.ALL_FORMATS).b
uild()

            cameraSource = CameraSource.Builder(this, barcodeDetector)
                .setRequestedPreviewSize(1920, 1080)
                .setAutoFocusEnabled(true) //you should add this
feature
                .build()

            binding.cameraSurfaceView.getHolder().addCallback(object :
SurfaceHolder.Callback {
                @SuppressLint("MissingPermission")
                override fun surfaceCreated(holder: SurfaceHolder) {
                    try {
                        //Start preview after 1s delay
                        cameraSource.start(holder)
                    } catch (e: IOException) {
                        e.printStackTrace()
                    }
                }

                @SuppressLint("MissingPermission")
                override fun surfaceChanged(
                    holder: SurfaceHolder,
                    format: Int,
                    width: Int,
                    height: Int
                ) {
                    try {
                        cameraSource.start(holder)
                    } catch (e: IOException) {
                        e.printStackTrace()
                    }
                }

                override fun surfaceDestroyed(holder: SurfaceHolder) {
                    cameraSource.stop()
                }
            })
```

Figure 7.2

```
            barcodeDetector.setProcessor(object :
Detector.Processor<Barcode> {
                override fun release() {
                    Toast.makeText(applicationContext, "Scanner has
been closed", Toast.LENGTH_SHORT)
                            .show()
                }

                override fun receiveDetections(detections:
Detections<Barcode>) {
                    val barcodes = detections.detectedItems
                    if (barcodes.size() == 1) {
                        scannedValue = barcodes.valueAt(0).rawValue


                        //Don't forget to add this line printing value
or finishing activity must run on main thread
                        runOnUiThread {
                            cameraSource.stop()
                            Toast.makeText(this@MainActivity, "value-
$scannedValue", Toast.LENGTH_SHORT).show()
                             finish()
                        }
                    }else
                    {
                        Toast.makeText(this@MainActivity, "value-
else", Toast.LENGTH_SHORT).show()

                    }
                }
            })
        }
```

```
        override fun onRequestPermissionsResult(
            requestCode: Int,
            permissions: Array<out String>,
            grantResults: IntArray
        ) {
            super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
            if (requestCode == requestCodeCameraPermission &&
grantResults.isNotEmpty()) {
                if (grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    setupControls()
                } else {
                    Toast.makeText(applicationContext, "Permission
Denied", Toast.LENGTH_SHORT).show()
                }
            }
        }

        override fun onDestroy() {
            super.onDestroy()
            cameraSource.stop()
        }
    }
```

Figure 7.3

CHAPTER FOUR

Chapter Four

Conclusion and future work

Attendance management is important to everyone Organizations such as educational Institutions.

Can Manage and control the success of any organization through Track people within The organization like students to maximize their performance.

The proposal the system introduces the process of monitoring the students present, Aims to help the teacher in the classroom or Laboratories to manage and record student attendance

Online and directly without the need to register Paper therefore will save time and Effort. System can Analyze data and display statistics about Student Absence, Absence Report Printing Percentages and student warnings about the limiter a period.

The developed system is easy to use and easy to use Which has an attractive and Simple GUI is design so that inserting, deleting, and changing data can be done easily Without interacting with tables in order to Increase the use of the application and make it easy to use and attractive The test case of the application revealed that a file The system works exciting and is ready to use Manage student attendance for any of the departments University, college or institute. Since our system is modular and can be extended effortlessly, future business ambitions are to make The system takes Attendance in other ways such as the face Recognize and use biometrics (fingerprinting) or mobile devices with NFC or RFID systems Moreover, we would like to make order Managing and recording attendance of employees University

References

1. XML – Managing Data Exchange by Wikibooks Copyright 2001, 2002 Elliotte Rusty Harold
2. DocBook: The Definitive Guide by Norman Walsh 2004-2021 Norman Walsh. Portions copyright © 1999-2010 O'Reilly Media, Inc.
3. Head First Kotlin: A Brain-Friendly Guide 1st Edition, Kindle Edition
4. https://www.apachefriends.org/faq_windows.html
5. https://www.apachefriends.org/index.html
6. https://webflow.com/blog/from-adobe-xd-to-webflow-how-to-turn-yourprototypes-into-live-websites
7. https://webflow.com/blog/from-sketch-to-webflow-how-to-turn-mockupsinto-live-websites
8. https://developer.android.com/training/basics/firstapp
9. https://www.w3schools.com/xml/
10. https://www.tutorialspoint.com/kotlin/index.htm
11. https://www.bluestacks.com/download.html
12. https://www.diagrams.net/blog/features.html