# COMP1786 – Mobile Application Design and Development Report

| **Your name** | **Bui Duy Hung** | **Your Student ID** | **001407425** |
| --- | --- | --- | --- |

# Brief statement of features you have implemented (2%)

|  |  |  |
| --- | --- | --- |
| **Feature** | **Status** | **Your Comments** |
| **Functionality A** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | The application meets the requirements for fields such as:   * Day of the week (e.g. Monday, Tuesday) - Required field * Time of course (e.g. 10:00, 11:00) - Required field * Capacity (how many persons can attend) - Required field * Duration (e.g. 60 minutes) - Required field * Price per class (e.g. £10) - Required field * Type of class (Flow Yoga, Aerial Yoga, Family Yoga) - Required field * Description   These fields are required to be filled in by the user otherwise they will be prompted with an error message. Any field that is not required will not show an error and will be saved in the database as null. |
| **Functionality B** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | When new data is added, it will be immediately saved to the SQLite database if the data is completely valid. In addition, when updating or deleting data, the SQLite database will change immediately. Finally, if the user wants to refresh the database, he will be able to choose to refresh the database of Class Instance or Course or both. |
| **Functionality C** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | When creating a course, you can create 1 or more classes inside. The fields of that class include:   * Date (e.g. 14/11/2024 – the date should match the day of the week – in the case of 14th of November, it would have to be a course running on Thursday) - Required field * Teacher (who is teaching this class) – Required field * Additional comments - Optional field   The added classes will be assigned a suitable course\_id code chosen by the user and the data is added, edited, deleted normally and saved in the SQLite database. |
| **Functionality D** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | The author has not implemented |
| **Functionality E** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | The author has not implemented |
| **Functionality F** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | The author has not implemented |
| **Functionality G** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | The author has not implemented |
| **Functionality H** | Fully completed  Partially completed  Having bugs/Not working  Not implemented | The author has not implemented |

|  |
| --- |
| **Link to recorded video (if you record your application before submitting the report)**  The recorded video will demonstrate the implemented product. It is roughly 10 minutes. |
| [Video CourseWork App](https://drive.google.com/file/d/132eFci_y1w4rUTXfvlMAE8JXu9xk5LS2/view?usp=drive_link) |

# SECTION 2 - REFLECTION (4%)

Write a reflection (approximately 350 words) on how the Apps, both Android and crossed platform, were developed. Discuss lessons learnt, what you think went well and what you think could have been improved and how.

I gained a lot of knowledge about application development from this project, including how to generate ideas, design products, code, and optimize systems for ease of maintenance. The tool that helps me create a fantastic application is Android Java. It offers features and libraries of its own that make building mobile applications more efficient. Among its benefits is a sizable community that enables me to swiftly access a wealth of information. It also provides reliable performance and strong memory management, and it can run on a wide variety of systems, which makes cross-platform application creation easier. However, I ran into issues with extended syntax when I was developing the aforementioned program, which resulted in larger code for the same purpose. That results in simple mistakes, particularly when writing intricate code. I must thoroughly and thoroughly learn the syntax and system in order to be able to use them more efficiently and get past these limitations. In addition, I will learn more about software architectural models such as MVC, MVVM, ... to help manage projects with a more modern structure and easier to maintain. Along with that, focus on the user interface to ensure that users feel easier to use and more eye-catching. Performance management is also a factor that helps users feel more comfortable when using the application without jerking or lagging. In summary, with a sizable global support network, Android Java remains a solid option for creating cross-platform and mobile applications.

# SECTION 3 - EVALUATION (10%):

An evaluation of your app(s). Write between 700 to 1000 words evaluating the app(s) that you have produced. Be specific and justify any statements you make. Your evaluation should include, but **need not be limited** to, the following aspects of your app:

i. Human computer interaction

ii. Security

iii. Ability of the app to run on a range of screen sizes and how this could be improved

iv. Changes that would need to be made in order for the app(s) to be deployed for live use

v. Others

The Yoga App I developed also needed to consider factors like human-computer interaction, security, how the app can run on different screen sizes and how to improve this, and what changes need to be made so that the app can be deployed for live use. Here are the items to consider:

a. Human computer interaction

HCI is a field of study that focuses on designing and developing computer systems so that users can interact with them easily, efficiently, and satisfyingly. In application development, HCI plays an extremely important role, directly affecting the user experience (UX) and the success of the product.

In this project, I decided to design a simple UI with all the features available on the screen for users to operate more conveniently. In addition, the data list display feature is optimized through "RecyclerView" to help display data more scientifically and modernly than the traditional "ListView":

A screenshot of a yoga course

Description automatically generated

Figure : UI App 1

In addition, when the user enters, there will be a successful entry message or an error message such as no information entered,...

A screenshot of a phone

Description automatically generated

Figure : Validation

b. Security

Android application security is extremely important to protect user data and maintain trust. In my opinion, applications should have the following security elements:

* Authentication and Authorization: Use multi-factor authentication methods (e.g. passwords, fingerprints, facial recognition) to enhance security. Also, grant users only the access they need to do their job and passwords should use strong encryption algorithms to store passwords and avoid storing passwords in clear text.
* Data Encryption: Encrypt sensitive data such as personal information,... before storing it on the device or server. When developing, use proven encryption libraries to ensure security.
* Source Code Protection: Do not hardcode sensitive information directly into the code.
* Update and Patch: Monitor security updates, always update libraries, frameworks and Android operating system to the latest version to patch security vulnerabilities.

These are the elements of application security that become perfect and create absolute trust for users.

c. Ability of the app to run on a range of screen sizes and how this could be improved

This affects the user experience to some extent. A good application will provide users with ease of use and make the most of the screen space, thereby attracting more users. Factors that affect adaptability include layout, font size, images, resources, etc. Accompanying factors such as using relatively appropriate measurement units, creating layouts for different screen sizes, testing on many different devices for flexible modifications, etc. This is a process that requires carefulness and thoroughness. By applying the right techniques and tools, it is possible to create applications with great user experiences on all devices.

d. Changes that would need to be made in order for the app(s) to be deployed for live use

Once you have completed building a simple Android application, you will want to share it or install it yourself to experience it. To be able to deploy the application, you need to perform the following steps:

* Testing and Debugging: Thorough testing ensures the application is free of syntax errors, logic errors, or user interface issues. Also test on multiple devices to ensure compatibility.
* Install system related packages: install them as they are required to install the application on the device.
* Device configuration: configure the device appropriately to be able to use the application more effectively.

e. Others

To develop more efficient applications, it is necessary to improve factors such as performance optimization, layout optimization, appropriate image loading, security, testing, updating and maintenance, reasonable user interface design, reasonable use of tools and libraries, clean code traces, applying patterns such as MVC, MVVP and dependency management. By combining the above factors, it is possible to develop high-quality Android applications that better meet the needs of users.

# SECTION 4 - DESIGN (2%)

Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.

A screenshot of a yoga course

Description automatically generated

Figure : Main App

A screenshot of a phone

Description automatically generated

Figure : Add course

A screenshot of a phone

Description automatically generated

Figure : Update Course

A person doing yoga on a rock

Description automatically generated

Figure : Main UI Class

A screenshot of a cell phone

Description automatically generated

Figure :Add class

A screenshot of a phone

Description automatically generated

Figure : Edit class

A screenshot of a yoga course

Description automatically generated

Figure : Delete data of Course or Class in DB

# SECTION 5 – CODE (2%)

|  |
| --- |
| Code listing of any code files you have written. You do not need to include generated code. Please clearly label the code, so it indicates the source file and programming language. |

MainActivity.java

public class MainActivity extends AppCompatActivity {  
  
  
 private AlertDialog.Builder dialogBuilder;  
 private AlertDialog dialog;  
 private CourseDatabaseHandle dbCourse;  
 private ClassDatabaseHandle dbClass;  
  
 private Spinner spinnerTypeYoga;  
 ArrayAdapter adapter2;  
  
 private Spinner spinnerDay;  
 ArrayAdapter adapter;  
  
// private EditText addDayYoga;  
 private EditText addPriceYoga;  
  
 private Spinner spinnerTime;  
 ArrayAdapter adapter1;  
  
 private EditText addCapacityYoga;  
 private EditText addDurationYoga;  
 private EditText addDescriptionYoga;  
  
 private Button saveButton;  
  
  
  
 private RecyclerView recyclerCoursesView;  
 private MainRecyclerViewAdapter mainRecyclerViewAdapter;  
 private List<Course> courseList;  
 private List<Course> courseListEdit;  
  
 private AppBarConfiguration appBarConfiguration;  
 private ActivityMainBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 setSupportActionBar(binding.toolbar);  
  
 binding.fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
// Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG)  
// .setAnchorView(R.id.fab)  
// .setAction("Action", null).show();  
  
 createAddCoursePopupDialog();  
 }  
 });  
  
 // Initialize  
 recyclerCoursesView = (RecyclerView) findViewById(R.id.*recyclerCoursesView*);  
 courseList = new ArrayList<>();  
 courseListEdit = new ArrayList<>();  
 dbCourse = new CourseDatabaseHandle(this);  
 dbClass = new ClassDatabaseHandle(this);  
  
  
 // Show list courses  
 recyclerCoursesView.setHasFixedSize(true);  
 recyclerCoursesView.setLayoutManager(new LinearLayoutManager(this));  
  
 courseList = dbCourse.getAllCourses();  
  
 for (Course c : courseList)  
 {  
 Course course = new Course();  
 course.setId(c.getId());  
 course.setTypeYoga(c.getTypeYoga());  
 course.setDayYoga(c.getDayYoga());  
 course.setPriceYoga(c.getPriceYoga() + "$");  
 course.setTimeYoga(c.getTimeYoga());  
 course.setCapacityYoga(c.getCapacityYoga());  
 course.setDurationYoga(c.getDurationYoga() + " mins");  
 course.setDescriptionYoga(c.getDescriptionYoga());  
 courseListEdit.add(course);  
 }  
  
 mainRecyclerViewAdapter = new MainRecyclerViewAdapter(this, courseListEdit);  
 recyclerCoursesView.setAdapter(mainRecyclerViewAdapter);  
 mainRecyclerViewAdapter.notifyDataSetChanged();  
  
  
 }  
  
 private void createAddCoursePopupDialog()  
 {  
 dialogBuilder = new AlertDialog.Builder(this);  
 View view = getLayoutInflater().inflate(R.layout.*add\_course*, null);  
  
 spinnerTypeYoga = (Spinner) view.findViewById(R.id.*spinnerTypeYoga*);  
 adapter2 = ArrayAdapter.*createFromResource*(this, R.array.*spinnerTypeYoga*, android.R.layout.*simple\_spinner\_item*);  
 adapter2.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 spinnerTypeYoga.setAdapter(adapter2);  
  
 spinnerDay = (Spinner) view.findViewById(R.id.*spinnerDay*);  
 adapter = ArrayAdapter.*createFromResource*(this, R.array.*spinnerDay*, android.R.layout.*simple\_spinner\_item*);  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 spinnerDay.setAdapter(adapter);  
  
 addPriceYoga = (EditText) view.findViewById(R.id.*addPriceYoga*);  
  
 spinnerTime = (Spinner) view.findViewById(R.id.*spinnerTime*);  
 adapter1 = ArrayAdapter.*createFromResource*(this, R.array.*spinnerTime*, android.R.layout.*simple\_spinner\_item*);  
 adapter1.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 spinnerTime.setAdapter(adapter1);  
  
 addCapacityYoga = (EditText) view.findViewById(R.id.*addCapacityYoga*);  
 addDurationYoga = (EditText) view.findViewById(R.id.*addDurationYoga*);  
 addDescriptionYoga = (EditText) view.findViewById(R.id.*addDescriptionYoga*);  
 saveButton = (Button) view.findViewById(R.id.*saveButton*);  
  
  
  
 dialogBuilder.setView(view);  
 dialog = dialogBuilder.create();  
 dialog.show();  
  
 saveButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 String priceYoga = addPriceYoga.getText().toString();  
 String capacityYoga = addCapacityYoga.getText().toString();  
 String durationYoga = addDurationYoga.getText().toString();  
 String descriptionYoga = addDescriptionYoga.getText().toString();  
  
 // Validate user input  
 boolean check = validateInfo( priceYoga,  
 capacityYoga, durationYoga, descriptionYoga);  
 if (check) {  
 saveYogaCourseToDB(v);  
 }  
 else  
 {  
 Toast.*makeText*(MainActivity.this, "Sorry, Check the Information", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private Boolean validateInfo(String priceYoga, String capacityYoga, String durationYoga, String descriptionYoga)  
 {  
 if (priceYoga.length() == 0)  
 {  
 addPriceYoga.requestFocus();  
 addPriceYoga.setError("Please enter price");  
 return false;  
 }  
 else if (capacityYoga.length() == 0) {  
 addCapacityYoga.requestFocus();  
 addCapacityYoga.setError("Please enter capacity");  
 return false;  
 }  
 else if (durationYoga.length() == 0) {  
 addDurationYoga.requestFocus();  
 addDurationYoga.setError("Please enter duration");  
 return false;  
 }  
 else if (descriptionYoga.length() == 0) {  
 addDescriptionYoga.requestFocus();  
 addDescriptionYoga.setError("Please enter description");  
 return false;  
 }  
 return true;  
 }  
  
 });  
 }  
  
 private void saveYogaCourseToDB(View v)  
 {  
 Course course = new Course();  
 String newAddTypeYoga = spinnerTypeYoga.getSelectedItem().toString();  
 String newAddDayYoga = spinnerDay.getSelectedItem().toString();  
// String newAddDayYoga = addDayYoga.getText().toString();  
 String newAddPriceYoga = addPriceYoga.getText().toString();  
 String newAddTimeYoga = spinnerTime.getSelectedItem().toString();  
 String newAddCapacityYoga = addCapacityYoga.getText().toString();  
 String newAddDurationYoga = addDurationYoga.getText().toString();  
 String newAddDescriptionYoga = addDescriptionYoga.getText().toString();  
  
 course.setTypeYoga(newAddTypeYoga);  
 course.setDayYoga(newAddDayYoga);  
 course.setPriceYoga(newAddPriceYoga);  
 course.setTimeYoga(newAddTimeYoga);  
 course.setCapacityYoga(newAddCapacityYoga);  
 course.setDurationYoga(newAddDurationYoga);  
 course.setDescriptionYoga(newAddDescriptionYoga);  
  
 // Save to db  
 dbCourse.addCourse(course);  
 Snackbar.*make*(v, "Course saved", Snackbar.*LENGTH\_LONG*).show();  
  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 dialog.dismiss();  
 startActivity(new Intent(MainActivity.this, MainActivity.class));  
 finish();  
 }  
 }, 1000);  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 getMenuInflater().inflate(R.menu.*menu\_main*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
 if (id == R.id.*delete\_course\_db*) {  
 dbCourse.deleteDataCourseTable();  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 dialog.dismiss();  
 startActivity(new Intent(MainActivity.this, MainActivity.class));  
 finish();  
 }  
 }, 1000);  
 return true;  
 }  
  
 if (id == R.id.*delete\_class\_db*)  
 {  
 dbClass.deleteDataClassTable();  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 dialog.dismiss();  
 startActivity(new Intent(MainActivity.this, MainActivity.class));  
 finish();  
 }  
 }, 1000);  
 return true;  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
}

DetailCourseActivity.java

public class DetailCourseActivity extends AppCompatActivity {  
  
 private TextView detailTypeYoga;  
 private TextView detailDayYoga;  
 private TextView detailPriceYoga;  
 private TextView detailTimeYoga;  
 private TextView detailCapacityYoga;  
 private TextView detailDurationYoga;  
 private TextView detailDescriptionYoga;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_detail\_course*);  
  
 detailTypeYoga = (TextView) findViewById(R.id.*detailTypeYoga*);  
 detailDayYoga = (TextView) findViewById(R.id.*detailDayYoga*);  
 detailPriceYoga = (TextView) findViewById(R.id.*detailPriceYoga*);  
 detailTimeYoga = (TextView) findViewById(R.id.*detailTimeYoga*);  
 detailCapacityYoga = (TextView) findViewById(R.id.*detailCapacityYoga*);  
 detailDurationYoga = (TextView) findViewById(R.id.*detailDurationYoga*);  
 detailDescriptionYoga = (TextView) findViewById(R.id.*detailDescriptionYoga*);  
  
 Bundle bundle = getIntent().getExtras();  
  
 if (bundle != null)  
 {  
 String typeYoga = bundle.getString("detailTypeYoga");  
 String dayYoga = bundle.getString("detailDayYoga");  
 String priceYoga = bundle.getString("detailPriceYoga");  
 String timeYoga = bundle.getString("detailTimeYoga");  
 String capacityYoga = bundle.getString("detailCapacityYoga");  
 String durationYoga = bundle.getString("detailDurationYoga");  
 String descriptionYoga = bundle.getString("detailDescriptionYoga");  
  
 detailTypeYoga.setText("Type of Yoga: " + typeYoga);  
 detailDayYoga.setText("Day: " + dayYoga);  
 detailPriceYoga.setText("Price: " + priceYoga);  
 detailTimeYoga.setText("Time: " + timeYoga);  
 detailCapacityYoga.setText("Capacity: " + capacityYoga);  
 detailDurationYoga.setText("Duration: " + durationYoga);  
 detailDescriptionYoga.setText("Description: " + descriptionYoga);  
  
  
  
 }  
 }  
}

ClassInstanceActivity.java

public class ClassInstanceActivity extends AppCompatActivity {  
  
 private AlertDialog.Builder dialogBuilder;  
 private AlertDialog dialog;  
 private ClassDatabaseHandle db;  
  
 private TextView addDateClass;  
 private TextView addTeacherClass;  
 private TextView addCommentClass;  
  
 private Button saveButtonClass;  
  
 private RecyclerView recyclerClassView;  
 private ClassRecyclerViewAdapter classRecyclerViewAdapter;  
 private List<ClassYoga> classYogaList;  
 private List<ClassYoga> classYogaListEdit;  
  
 private AppBarConfiguration appBarConfiguration;  
 private ActivityClassInstanceBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
  
 int course\_id = 0;  
 Bundle bundle = getIntent().getExtras();  
 if (bundle != null) {  
 course\_id = bundle.getInt("course\_id");  
 }  
  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_class\_instance*);  
  
 binding = ActivityClassInstanceBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 setSupportActionBar(binding.toolbar);  
  
 binding.fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
// Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG)  
// .setAnchorView(R.id.fab)  
// .setAction("Action", null).show();  
  
 int course\_id = 0;  
 Bundle bundle = getIntent().getExtras();  
 if (bundle != null) {  
 course\_id = bundle.getInt("course\_id");  
 }  
  
 createAddClassPopupDialog(course\_id);  
 }  
 });  
  
 // Initialize  
 recyclerClassView = (RecyclerView) findViewById(R.id.*recyclerClassesView*);  
 classYogaList = new ArrayList<>();  
 classYogaListEdit = new ArrayList<>();  
 db = new ClassDatabaseHandle(this);  
  
 // Show list class  
 recyclerClassView.setHasFixedSize(true);  
 recyclerClassView.setLayoutManager(new LinearLayoutManager(this));  
  
  
 classYogaList = db.getAllClassYoga(course\_id);  
  
 for (ClassYoga c : classYogaList) {  
 ClassYoga classYoga = new ClassYoga();  
 classYoga.setId(c.getId());  
 classYoga.setCourse\_id(c.getCourse\_id());  
 classYoga.setClassDate(c.getClassDate());  
 classYoga.setTeacherName(c.getTeacherName());  
 classYoga.setComment(c.getComment());  
  
 classYogaListEdit.add(classYoga);  
 }  
  
 classRecyclerViewAdapter = new ClassRecyclerViewAdapter(this, classYogaListEdit);  
 recyclerClassView.setAdapter(classRecyclerViewAdapter);  
 classRecyclerViewAdapter.notifyDataSetChanged();  
  
 }  
  
 private void createAddClassPopupDialog(int course\_id) {  
 dialogBuilder = new AlertDialog.Builder(this);  
 View view = getLayoutInflater().inflate(R.layout.*add\_class*, null);  
  
 addDateClass = (EditText) view.findViewById(R.id.*addDateClass*);  
 addTeacherClass = (EditText) view.findViewById(R.id.*addTeacherClass*);  
 addCommentClass = (EditText) view.findViewById(R.id.*addCommentClass*);  
 saveButtonClass = (Button) view.findViewById(R.id.*saveButtonClass*);  
  
 final Calendar calendar = Calendar.*getInstance*();  
 final int year = calendar.get(Calendar.*YEAR*);  
 final int month = calendar.get(Calendar.*MONTH*);  
 final int day = calendar.get(Calendar.*DAY\_OF\_MONTH*);  
 addDateClass.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 DatePickerDialog datePickerDialog = new DatePickerDialog(ClassInstanceActivity.this, new DatePickerDialog.OnDateSetListener() {  
 @Override  
 public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {  
 CourseDatabaseHandle courseDB = new CourseDatabaseHandle(ClassInstanceActivity.this);  
 if (courseDB.getCourse(course\_id) != null)  
 {  
 Course course = courseDB.getCourse(course\_id);  
 String dayCourse = course.getDayYoga();  
 String day\_of\_the\_week\_class = DateUtil.*showDayOfTheWeek*(dayOfMonth, month + 1, year);  
 if(!day\_of\_the\_week\_class.equals(dayCourse)) {  
 Toast.*makeText*(ClassInstanceActivity.this, "The date does not match the day of the week!", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 @SuppressLint("DefaultLocale") String date = String.*format*("%02d/%02d/%d", dayOfMonth, month + 1, year);  
 Toast.*makeText*(ClassInstanceActivity.this, "Select date: " + date, Toast.*LENGTH\_SHORT*).show();  
 addDateClass.setText(date);  
 }  
 }  
 }, year, month, day);  
 datePickerDialog.show();  
 }  
 });  
  
  
 dialogBuilder.setView(view);  
 dialog = dialogBuilder.create();  
 dialog.show();  
  
  
 saveButtonClass.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 String dateClass = addDateClass.getText().toString().trim();  
 String teacherClass = addTeacherClass.getText().toString();  
  
 // Validate user input  
 boolean check = validateInfo(dateClass, teacherClass);  
 if (check) {  
 saveClassToDB(v, course\_id);  
 Toast.*makeText*(ClassInstanceActivity.this, "Saved Class to DB!", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(ClassInstanceActivity.this, "Sorry, Check the Information Class", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 }  
  
 private Boolean validateInfo(String dateClass, String teacherClass) {  
 if (dateClass.length() == 0) {  
 addDateClass.requestFocus();  
 addDateClass.setError("Please enter date");  
 return false;  
 } else if (teacherClass.length() == 0) {  
 addTeacherClass.requestFocus();  
 addTeacherClass.setError("Please enter teacher name");  
 return false;  
 }  
 return true;  
 }  
  
 });  
 }  
  
 private void saveClassToDB(View v, int course\_id) {  
 ClassYoga classYoga = new ClassYoga();  
  
 String newAddDateClass = addDateClass.getText().toString().trim();  
 String newAddTeacherClass = addTeacherClass.getText().toString();  
 String newAddCommentClass = addCommentClass.getText().toString();  
  
 classYoga.setCourse\_id(course\_id);  
 classYoga.setClassDate(newAddDateClass);  
 classYoga.setTeacherName(newAddTeacherClass);  
 classYoga.setComment(newAddCommentClass);  
  
 // Save to db  
 db.addClass(classYoga);  
 Snackbar.*make*(v, "Class saved", Snackbar.*LENGTH\_LONG*).show();  
  
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 dialog.dismiss();  
 startActivity(new Intent(ClassInstanceActivity.this, MainActivity.class));  
 finish();  
 }  
 }, 1000);  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 getMenuInflater().inflate(R.menu.*menu\_main*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
// if (id == R.id.action\_settings) {  
// return true;  
// }  
  
 return super.onOptionsItemSelected(item);  
 }  
  
}

CourseDatabaseHandle.java

public class CourseDatabaseHandle extends SQLiteOpenHelper {  
  
 private Context ctx;  
  
 public CourseDatabaseHandle(@Nullable Context context) {  
 super(context, Constants.*COURSE\_DB\_NAME*, null, Constants.*DB\_VERSION*);  
 this.ctx = context;  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String CREATE\_YOGA\_COURSE\_TABLE = "CREATE TABLE " + Constants.*COURSE\_TABLE\_NAME* + "("  
 + Constants.*COURSE\_KEY\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT,"  
 + Constants.*COURSE\_KEY\_TYPE\_YOGA* + " TEXT,"  
 + Constants.*COURSE\_KEY\_DAY\_YOGA* + " TEXT,"  
 + Constants.*COURSE\_KEY\_PRICE\_YOGA* + " TEXT,"  
 + Constants.*COURSE\_KEY\_TIME\_YOGA* + " TEXT,"  
 + Constants.*COURSE\_KEY\_CAPACITY\_YOGA* + " TEXT,"  
 + Constants.*COURSE\_KEY\_DURATION\_YOGA* + " TEXT,"  
 + Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA* + " TEXT"  
 + ")";  
  
 db.execSQL(CREATE\_YOGA\_COURSE\_TABLE);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + Constants.*COURSE\_TABLE\_NAME*);  
 onCreate(db);  
 }  
  
 /\*  
 CRUD Operations (Create, Read, Update, Delete)  
 \*/  
  
 // Add course  
 public void addCourse(Course course)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(Constants.*COURSE\_KEY\_TYPE\_YOGA*, course.getTypeYoga());  
 values.put(Constants.*COURSE\_KEY\_DAY\_YOGA*, course.getDayYoga());  
 values.put(Constants.*COURSE\_KEY\_PRICE\_YOGA*, course.getPriceYoga());  
 values.put(Constants.*COURSE\_KEY\_TIME\_YOGA*, course.getTimeYoga());  
 values.put(Constants.*COURSE\_KEY\_CAPACITY\_YOGA*, course.getCapacityYoga());  
 values.put(Constants.*COURSE\_KEY\_DURATION\_YOGA*, course.getDurationYoga());  
 values.put(Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA*, course.getDescriptionYoga());  
  
 // Insert the row  
 db.insert(Constants.*COURSE\_TABLE\_NAME*, null, values);  
  
 Log.*d*("Saved!!", "Saved to DB");  
 Log.*d*("Type: ", course.getTypeYoga());  
 Log.*d*("Day: ", course.getDayYoga());  
 Log.*d*("Price: ", course.getPriceYoga());  
 Log.*d*("Time: ", course.getTimeYoga());  
 Log.*d*("Capacity: ", course.getCapacityYoga());  
 Log.*d*("Duration: ", course.getDurationYoga());  
 Log.*d*("Description: ", course.getDescriptionYoga());  
 }  
  
 // Get a course  
 public Course getCourse(int id)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
  
 Cursor cursor = db.query(Constants.*COURSE\_TABLE\_NAME*, new String[] {  
 Constants.*COURSE\_KEY\_ID*,  
 Constants.*COURSE\_KEY\_TYPE\_YOGA*,  
 Constants.*COURSE\_KEY\_DAY\_YOGA*,  
 Constants.*COURSE\_KEY\_PRICE\_YOGA*,  
 Constants.*COURSE\_KEY\_TIME\_YOGA*,  
 Constants.*COURSE\_KEY\_CAPACITY\_YOGA*,  
 Constants.*COURSE\_KEY\_DURATION\_YOGA*,  
 Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA*},  
 Constants.*COURSE\_KEY\_ID* + "=?",  
 new String[] {String.*valueOf*(id)}, null, null, null, null);  
  
 Course course = new Course();  
 if (cursor != null)  
 {  
 cursor.moveToFirst();  
  
 // Get course  
 int idIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_ID*);  
 int typeYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_TYPE\_YOGA*);  
 int dayYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_DAY\_YOGA*);  
 int priceYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_PRICE\_YOGA*);  
 int timeYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_TIME\_YOGA*);  
 int capacityYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_CAPACITY\_YOGA*);  
 int durationYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_DURATION\_YOGA*);  
 int descriptionYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA*);  
 if (idIndex >= 0 && typeYogaIndex >= 0 && priceYogaIndex >= 0 && timeYogaIndex >= 0  
 && capacityYogaIndex >= 0 && durationYogaIndex >= 0 && descriptionYogaIndex >= 0)  
 {  
 course.setId(Integer.*parseInt*(cursor.getString(idIndex)));  
 course.setTypeYoga(cursor.getString(typeYogaIndex));  
 course.setDayYoga(cursor.getString(dayYogaIndex));  
 course.setPriceYoga(cursor.getString(priceYogaIndex));  
 course.setTimeYoga(cursor.getString(timeYogaIndex));  
 course.setCapacityYoga(cursor.getString(capacityYogaIndex));  
 course.setDurationYoga(cursor.getString(durationYogaIndex));  
 course.setDescriptionYoga(cursor.getString(descriptionYogaIndex));  
 }  
 }  
 cursor.close();  
 return course;  
 }  
  
 // Get all courses  
 public List<Course> getAllCourses()  
 {  
 SQLiteDatabase db = this.getReadableDatabase();  
  
 List<Course> courseList = new ArrayList<>();  
  
 Cursor cursor = db.query(Constants.*COURSE\_TABLE\_NAME*, new String[] {  
 Constants.*COURSE\_KEY\_ID*,  
 Constants.*COURSE\_KEY\_TYPE\_YOGA*,  
 Constants.*COURSE\_KEY\_DAY\_YOGA*,  
 Constants.*COURSE\_KEY\_PRICE\_YOGA*,  
 Constants.*COURSE\_KEY\_TIME\_YOGA*,  
 Constants.*COURSE\_KEY\_CAPACITY\_YOGA*,  
 Constants.*COURSE\_KEY\_DURATION\_YOGA*,  
 Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA*},  
 null, null, null, null, Constants.*COURSE\_KEY\_ID* + " ASC", null); // ASC: ascending, DESC: descending  
  
// Cursor cursor = db.query(Constants.COURSE\_TABLE\_NAME, null,  
// null, null, null, null, null, null); // ASC: ascending, DESC: descending  
  
 if (cursor.moveToFirst())  
 {  
 do {  
 Course course = new Course();  
 int idIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_ID*);  
 int typeYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_TYPE\_YOGA*);  
 int dayYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_DAY\_YOGA*);  
 int priceYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_PRICE\_YOGA*);  
 int timeYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_TIME\_YOGA*);  
 int capacityYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_CAPACITY\_YOGA*);  
 int durationYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_DURATION\_YOGA*);  
 int descriptionYogaIndex = cursor.getColumnIndex(Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA*);  
 if (idIndex >= 0 && typeYogaIndex >= 0 && priceYogaIndex >= 0 && timeYogaIndex >= 0  
 && capacityYogaIndex >= 0 && durationYogaIndex >= 0 && descriptionYogaIndex >= 0)  
 {  
 course.setId(Integer.*parseInt*(cursor.getString(idIndex)));  
 course.setTypeYoga(cursor.getString(typeYogaIndex));  
 course.setDayYoga(cursor.getString(dayYogaIndex));  
 course.setPriceYoga(cursor.getString(priceYogaIndex));  
 course.setTimeYoga(cursor.getString(timeYogaIndex));  
 course.setCapacityYoga(cursor.getString(capacityYogaIndex));  
 course.setDurationYoga(cursor.getString(durationYogaIndex));  
 course.setDescriptionYoga(cursor.getString(descriptionYogaIndex));  
 }  
  
 courseList.add(course);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 return courseList;  
 }  
  
 // Update course  
 public int updateCourse(Course course)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(Constants.*COURSE\_KEY\_TYPE\_YOGA*, course.getTypeYoga());  
 values.put(Constants.*COURSE\_KEY\_DAY\_YOGA*, course.getDayYoga());  
 values.put(Constants.*COURSE\_KEY\_PRICE\_YOGA*, course.getPriceYoga());  
 values.put(Constants.*COURSE\_KEY\_TIME\_YOGA*, course.getTimeYoga());  
 values.put(Constants.*COURSE\_KEY\_CAPACITY\_YOGA*, course.getCapacityYoga());  
 values.put(Constants.*COURSE\_KEY\_DURATION\_YOGA*, course.getDurationYoga());  
 values.put(Constants.*COURSE\_KEY\_DESCRIPTION\_YOGA*, course.getDescriptionYoga());  
  
 // update row  
 return db.update(Constants.*COURSE\_TABLE\_NAME*, values, Constants.*COURSE\_KEY\_ID* + "=?",  
 new String[] {String.*valueOf*(course.getId())}); // return ID of row updated  
 }  
  
 // Delete course  
 public void deleteCourse(int id)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete(Constants.*COURSE\_TABLE\_NAME*, Constants.*COURSE\_KEY\_ID* + "=?",  
 new String[] {String.*valueOf*(id)});  
 db.close();  
 }  
  
 // Get count  
 public int getCourseCount()  
 {  
 String countQuery = "SELECT \* FROM " + Constants.*COURSE\_TABLE\_NAME*;  
 SQLiteDatabase db = this.getReadableDatabase();  
  
 Cursor cursor = db.rawQuery(countQuery, null); // rawQuery: Runs the provided SQL and returns a "Cursor" over the result set (multiple results).  
 cursor.close();  
 return cursor.getCount();  
 }  
  
 // Detele table  
 public void deleteDataCourseTable()  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete(Constants.*COURSE\_TABLE\_NAME*, null, null);  
 db.close();  
 }  
  
}

ClassDatabaseHandle.java

public class ClassDatabaseHandle extends SQLiteOpenHelper {  
  
 private final Context ctx;  
  
 public ClassDatabaseHandle(@Nullable Context context) {  
 super(context, Constants.*CLASS\_INSTANCE\_DB\_NAME*, null, Constants.*DB\_VERSION*);  
 this.ctx = context;  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String CREATE\_CLASS\_INSTANCE\_TABLE = "CREATE TABLE " + Constants.*CLASS\_INSTANCE\_TABLE\_NAME* + "(" +  
 Constants.*CLASS\_INSTANCE\_KEY\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT,"  
 + Constants.*CLASS\_INSTANCE\_KEY\_COURSE\_ID* + " INTEGER NOT NULL,"  
 + Constants.*CLASS\_INSTANCE\_KEY\_TEACHER\_NAME* + " TEXT,"  
 + Constants.*CLASS\_INSTANCE\_KEY\_CLASS\_DATE* + " TEXT,"  
 + Constants.*CLASS\_INSTANCE\_KEY\_COMMENT* + " TEXT,"  
// + "FOREIGN KEY(" + Constants.CLASS\_INSTANCE\_KEY\_COURSE\_ID + ") REFERENCES " + Constants.COURSE\_TABLE\_NAME + "(" + Constants.COURSE\_KEY\_ID + ")" + ")";  
 + "FOREIGN KEY(" + Constants.*CLASS\_INSTANCE\_KEY\_COURSE\_ID* + ") REFERENCES " + Constants.*COURSE\_TABLE\_NAME* + "(" + Constants.*COURSE\_KEY\_ID* + ")"  
 + ")";  
 db.execSQL(CREATE\_CLASS\_INSTANCE\_TABLE);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("DROP TABLE IF EXISTS " + Constants.*CLASS\_INSTANCE\_TABLE\_NAME*);  
 onCreate(db);  
 }  
  
 /\*  
 CRUD Operations (Create, Read, Update, Delete)  
 \*/  
  
 // Add class  
 public void addClass(ClassYoga classYoga)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_COURSE\_ID*, classYoga.getCourse\_id());  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_TEACHER\_NAME*, classYoga.getTeacherName());  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_CLASS\_DATE*, classYoga.getClassDate());  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_COMMENT*, classYoga.getComment());  
  
 // Insert the row  
 db.insert(Constants.*CLASS\_INSTANCE\_TABLE\_NAME*, null, values);  
  
 Log.*d*("Saved!!", "Saved to DB");  
 Log.*d*("Teacher Name: ", classYoga.getTeacherName());  
 Log.*d*("Class Date: ", classYoga.getClassDate());  
 Log.*d*("Comment: ", classYoga.getComment());  
  
 db.close();  
 }  
  
 // Get a class  
// public ClassYoga getClassYoga(int id)  
// {  
// SQLiteDatabase db = this.getWritableDatabase();  
//  
// Cursor cursor = db.query(Constants.CLASS\_INSTANCE\_TABLE\_NAME, new String[] {  
// Constants.CLASS\_INSTANCE\_KEY\_ID,  
// Constants.CLASS\_INSTANCE\_KEY\_COURSE\_ID,  
// Constants.CLASS\_INSTANCE\_KEY\_TEACHER\_NAME,  
// Constants.CLASS\_INSTANCE\_KEY\_CLASS\_DATE,  
// Constants.CLASS\_INSTANCE\_KEY\_COMMENT},  
// Constants.CLASS\_INSTANCE\_KEY\_ID + "=?",  
// new String[] {String.valueOf(id)}, null, null, null, null);  
//  
// ClassYoga classYoga = new ClassYoga();  
// if (cursor != null)  
// {  
// cursor.moveToFirst();  
//  
// // Get class  
// int idIndex = cursor.getColumnIndex(Constants.CLASS\_INSTANCE\_KEY\_ID);  
// int courseIDIndex = cursor.getColumnIndex(Constants.CLASS\_INSTANCE\_KEY\_COURSE\_ID);  
// int teacherNameIndex = cursor.getColumnIndex(Constants.CLASS\_INSTANCE\_KEY\_TEACHER\_NAME);  
// int dateClassIndex = cursor.getColumnIndex(Constants.CLASS\_INSTANCE\_KEY\_CLASS\_DATE);  
// int commentClassIndex = cursor.getColumnIndex(Constants.CLASS\_INSTANCE\_KEY\_COMMENT);  
//  
// if (idIndex >= 0 && courseIDIndex >= 0 && teacherNameIndex >= 0 && dateClassIndex >= 0  
// && commentClassIndex >= 0 )  
// {  
// classYoga.setId(Integer.parseInt(cursor.getString(idIndex)));  
// classYoga.setCourse\_id(Integer.parseInt(cursor.getString(courseIDIndex)));  
// classYoga.setTeacherName(cursor.getString(teacherNameIndex));  
// classYoga.setClassDate(cursor.getString(dateClassIndex));  
// classYoga.setComment(cursor.getString(commentClassIndex));  
// }  
// }  
// cursor.close();  
// return classYoga;  
// }  
  
 // Get all class  
 public List<ClassYoga> getAllClassYoga(int course\_id)  
 {  
 SQLiteDatabase db = this.getReadableDatabase();  
  
 List<ClassYoga> classYogaList = new ArrayList<>();  
  
 Cursor cursor = db.rawQuery("SELECT \* FROM " + Constants.*CLASS\_INSTANCE\_TABLE\_NAME* +  
 " WHERE " + Constants.*CLASS\_INSTANCE\_KEY\_COURSE\_ID* + " = ?", new String[] {String.*valueOf*(course\_id)}); // ASC: ascending, DESC: descending  
  
// Cursor cursor = db.query(Constants.COURSE\_TABLE\_NAME, null,  
// null, null, null, null, null, null); // ASC: ascending, DESC: descending  
  
 if (cursor.moveToFirst())  
 {  
 do {  
 ClassYoga classYoga = new ClassYoga();  
 int idIndex = cursor.getColumnIndex(Constants.*CLASS\_INSTANCE\_KEY\_ID*);  
 int courseIDIndex = cursor.getColumnIndex(Constants.*CLASS\_INSTANCE\_KEY\_COURSE\_ID*);  
 int teacherNameIndex = cursor.getColumnIndex(Constants.*CLASS\_INSTANCE\_KEY\_TEACHER\_NAME*);  
 int dateClassIndex = cursor.getColumnIndex(Constants.*CLASS\_INSTANCE\_KEY\_CLASS\_DATE*);  
 int commentClassIndex = cursor.getColumnIndex(Constants.*CLASS\_INSTANCE\_KEY\_COMMENT*);  
 if (idIndex >= 0  
 && courseIDIndex >= 0  
 && teacherNameIndex >= 0 && dateClassIndex >= 0  
 && commentClassIndex >= 0 )  
 {  
 classYoga.setId(Integer.*parseInt*(cursor.getString(idIndex)));  
 classYoga.setCourse\_id(Integer.*parseInt*(cursor.getString(courseIDIndex)));  
 classYoga.setTeacherName(cursor.getString(teacherNameIndex));  
 classYoga.setClassDate(cursor.getString(dateClassIndex));  
 classYoga.setComment(cursor.getString(commentClassIndex));  
 }  
  
 classYogaList.add(classYoga);  
 } while (cursor.moveToNext());  
 }  
 cursor.close();  
 return classYogaList;  
 }  
  
 // Update class  
 public int updateClassYoga(ClassYoga classYoga)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
  
 ContentValues values = new ContentValues();  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_COURSE\_ID*, classYoga.getCourse\_id());  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_TEACHER\_NAME*, classYoga.getTeacherName());  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_CLASS\_DATE*, classYoga.getClassDate());  
 values.put(Constants.*CLASS\_INSTANCE\_KEY\_COMMENT*, classYoga.getComment());  
  
 // update row  
 return db.update(Constants.*CLASS\_INSTANCE\_TABLE\_NAME*, values, Constants.*CLASS\_INSTANCE\_KEY\_ID* + "=?",  
 new String[] {String.*valueOf*(classYoga.getId())}); // return ID of row updated  
 }  
  
 // Delete class  
 public void deleteClassYoga(int id)  
 {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete(Constants.*CLASS\_INSTANCE\_TABLE\_NAME*, Constants.*CLASS\_INSTANCE\_KEY\_ID* + "=?",  
 new String[] {String.*valueOf*(id)});  
 db.close();  
 }  
  
 // Get count  
 public int getClassCount()  
 {  
 String countQuery = "SELECT \* FROM " + Constants.*CLASS\_INSTANCE\_TABLE\_NAME*;  
 SQLiteDatabase db = this.getReadableDatabase();  
  
 Cursor cursor = db.rawQuery(countQuery, null); // rawQuery: Runs the provided SQL and returns a "Cursor" over the result set (multiple results).  
 cursor.close();  
 return cursor.getCount();  
 }  
  
 // Delete data class  
 public void deleteDataClassTable() {  
 SQLiteDatabase db = this.getWritableDatabase();  
 db.delete(Constants.*CLASS\_INSTANCE\_TABLE\_NAME*, null, null);  
 db.close();  
 }  
  
}

Course.java

public class Course  
{  
 private int id;  
 private String typeYoga;  
 private String dayYoga;  
 private String priceYoga;  
 private String timeYoga;  
 private String capacityYoga;  
 private String durationYoga;  
 private String descriptionYoga;  
  
 public Course() {  
 }  
  
 public Course(int id, String typeYoga, String dayYoga, String priceYoga,  
 String timeYoga, String capacityYoga, String durationYoga, String descriptionYoga) {  
 this.id = id;  
 this.typeYoga = typeYoga;  
 this.dayYoga = dayYoga;  
 this.priceYoga = priceYoga;  
 this.timeYoga = timeYoga;  
 this.capacityYoga = capacityYoga;  
 this.durationYoga = durationYoga;  
 this.descriptionYoga = descriptionYoga;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getTypeYoga() {  
 return typeYoga;  
 }  
  
 public void setTypeYoga(String typeYoga) {  
 this.typeYoga = typeYoga;  
 }  
  
 public String getDayYoga() {  
 return dayYoga;  
 }  
  
 public void setDayYoga(String dayYoga) {  
 this.dayYoga = dayYoga;  
 }  
  
 public String getPriceYoga() {  
 return priceYoga;  
 }  
  
 public void setPriceYoga(String priceYoga) {  
 this.priceYoga = priceYoga;  
 }  
  
 public String getTimeYoga() {  
 return timeYoga;  
 }  
  
 public void setTimeYoga(String timeYoga) {  
 this.timeYoga = timeYoga;  
 }  
  
 public String getCapacityYoga() {  
 return capacityYoga;  
 }  
  
 public void setCapacityYoga(String capacityYoga) {  
 this.capacityYoga = capacityYoga;  
 }  
  
 public String getDurationYoga() {  
 return durationYoga;  
 }  
  
 public void setDurationYoga(String durationYoga) {  
 this.durationYoga = durationYoga;  
 }  
  
 public String getDescriptionYoga() {  
 return descriptionYoga;  
 }  
  
 public void setDescriptionYoga(String descriptionYoga) {  
 this.descriptionYoga = descriptionYoga;  
 }  
}

ClassYoga.java

public class ClassYoga  
{  
 private int id;  
 private int course\_id;  
 private String teacherName;  
 private String classDate;  
 private String comment;  
  
 public ClassYoga() {  
 }  
  
 public ClassYoga(int id, int course\_id, String teacherName, String classDate, String comment) {  
 this.id = id;  
 this.course\_id = course\_id;  
 this.teacherName = teacherName;  
 this.classDate = classDate;  
 this.comment = comment;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public int getCourse\_id() {  
 return course\_id;  
 }  
  
 public void setCourse\_id(int course\_id) {  
 this.course\_id = course\_id;  
 }  
  
 public String getTeacherName() {  
 return teacherName;  
 }  
  
 public void setTeacherName(String teacherName) {  
 this.teacherName = teacherName;  
 }  
  
 public String getClassDate() {  
 return classDate;  
 }  
  
 public void setClassDate(String classDate) {  
 this.classDate = classDate;  
 }  
  
 public String getComment() {  
 return comment;  
 }  
  
 public void setComment(String comment) {  
 this.comment = comment;  
 }  
}

MainRecyclerViewAdapter.java

public class MainRecyclerViewAdapter extends RecyclerView.Adapter<MainRecyclerViewAdapter.ViewHolder>  
{  
  
 private Context context;  
 private List<Course> courseList;  
 private LayoutInflater inflater;  
  
 private AlertDialog.Builder alertDialogBuilder;  
 private AlertDialog dialog;  
  
 public MainRecyclerViewAdapter(Context context, List<Course> courseLists) {  
 this.context = context;  
 this.courseList = courseLists;  
 }  
  
 @NonNull  
 @Override  
 public MainRecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*courses\_list*, parent, false);  
 return new ViewHolder(view, context);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull MainRecyclerViewAdapter.ViewHolder holder, int position) {  
 Course course = courseList.get(position);  
  
 holder.typeYoga.setText(course.getTypeYoga());  
 holder.dayYoga.setText(course.getDayYoga());  
 holder.priceYoga.setText(String.*valueOf*(course.getPriceYoga()));  
 holder.timeYoga.setText(course.getTimeYoga());  
 holder.capacityYoga.setText(course.getCapacityYoga());  
 holder.durationYoga.setText(course.getDurationYoga());  
 holder.descriptionYoga.setText(course.getDescriptionYoga());  
 }  
  
 @Override  
 public int getItemCount() {  
 return courseList.size();  
 }  
  
 public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener  
 {  
 private TextView typeYoga;  
 private TextView dayYoga;  
 private TextView priceYoga;  
 private TextView timeYoga;  
 private TextView capacityYoga;  
 private TextView durationYoga;  
 private TextView descriptionYoga;  
  
 private Button editButton;  
 private Button deleteButton;  
 private Button listClasses;  
  
  
  
 public ViewHolder(@NonNull View view, Context ctx) {  
 super(view);  
 context = ctx;  
  
 typeYoga = (TextView) view.findViewById(R.id.*typeYoga*);  
 dayYoga = (TextView) view.findViewById(R.id.*dayYoga*);  
 priceYoga = (TextView) view.findViewById(R.id.*priceYoga*);  
 timeYoga = (TextView) view.findViewById(R.id.*timeYoga*);  
 capacityYoga = (TextView) view.findViewById(R.id.*capacityYoga*);  
 durationYoga = (TextView) view.findViewById(R.id.*durationYoga*);  
 descriptionYoga = (TextView) view.findViewById(R.id.*descriptionYoga*);  
  
 editButton = (Button) view.findViewById(R.id.*editButton*);  
 deleteButton = (Button) view.findViewById(R.id.*deleteButton*);  
 listClasses = (Button) view.findViewById(R.id.*listClasses*);  
  
 editButton.setOnClickListener(this);  
 deleteButton.setOnClickListener(this);  
 listClasses.setOnClickListener(this);  
  
 view.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Go to next screen => DetailsActivity  
 int position = getAdapterPosition();  
 Course course = courseList.get(position);  
 Intent intent = new Intent(context, DetailCourseActivity.class);  
  
 intent.putExtra("detailTypeYoga", course.getTypeYoga());  
 intent.putExtra("detailDayYoga", course.getDayYoga());  
 intent.putExtra("detailPriceYoga", course.getPriceYoga());  
 intent.putExtra("detailTimeYoga", course.getTimeYoga());  
 intent.putExtra("detailCapacityYoga", course.getCapacityYoga());  
 intent.putExtra("detailDurationYoga", course.getDurationYoga());  
 intent.putExtra("detailDescriptionYoga", course.getDescriptionYoga());  
  
 context.startActivity(intent);  
 }  
 });  
  
 }  
  
  
  
 @Override  
 public void onClick(View v) {  
 int position = getAdapterPosition();  
 Course course = courseList.get(position);  
  
 if (v.getId() == R.id.*editButton*) {  
 editButton(course);  
 }  
  
 if (v.getId() == R.id.*deleteButton*) {  
 deleteButton(course.getId());  
 }  
  
 if (v.getId() == R.id.*listClasses*)  
 {  
 changeClassActivity(course.getId());  
 }  
 }  
  
 public void editButton(Course course)  
 {  
 alertDialogBuilder = new AlertDialog.Builder(context);  
  
 inflater = LayoutInflater.*from*(context);  
 View view = inflater.inflate(R.layout.*edit\_course*, null);  
  
 final TextView title = (TextView) view.findViewById(R.id.*titleEditYogaCourse*);  
 final Spinner editSpinnerTypeYoga = (Spinner) view.findViewById(R.id.*editSpinnerTypeYoga*);  
 ArrayAdapter adapter2;  
 final Spinner editSpinnerDay = (Spinner) view.findViewById(R.id.*editSpinnerDay*);  
 ArrayAdapter adapter;  
 final EditText editPriceYoga = (EditText) view.findViewById(R.id.*editPriceYoga*);  
 final Spinner editSpinnerTime = (Spinner) view.findViewById(R.id.*editSpinnerTime*);  
 ArrayAdapter adapter1;  
 final EditText editCapacityYoga = (EditText) view.findViewById(R.id.*editCapacityYoga*);  
 final EditText editDurationYoga = (EditText) view.findViewById(R.id.*editDurationYoga*);  
 final EditText editDescriptionYoga = (EditText) view.findViewById(R.id.*editDescriptionYoga*);  
  
  
 title.setText("Edit Course Yoga");  
 editPriceYoga.setText(course.getPriceYoga());  
 editCapacityYoga.setText(course.getCapacityYoga());  
 editDurationYoga.setText(course.getDurationYoga());  
 editDescriptionYoga.setText(course.getDescriptionYoga());  
  
 adapter2 = ArrayAdapter.*createFromResource*(context, R.array.*spinnerTypeYoga*, android.R.layout.*simple\_spinner\_item*);  
 adapter2.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 editSpinnerTypeYoga.setAdapter(adapter2);  
  
 adapter = ArrayAdapter.*createFromResource*(context, R.array.*spinnerDay*, android.R.layout.*simple\_spinner\_item*);  
 adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 editSpinnerDay.setAdapter(adapter);  
  
 adapter1 = ArrayAdapter.*createFromResource*(context, R.array.*spinnerTime*, android.R.layout.*simple\_spinner\_item*);  
 adapter1.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
 editSpinnerTime.setAdapter(adapter1);  
  
 Button saveButton = (Button) view.findViewById(R.id.*saveEditButton*);  
  
  
 alertDialogBuilder.setView(view);  
 dialog = alertDialogBuilder.create();  
 dialog.show();  
  
 saveButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 CourseDatabaseHandle db = new CourseDatabaseHandle(context);  
  
 // Update course  
 course.setTypeYoga(editSpinnerTypeYoga.getSelectedItem().toString());  
 course.setDayYoga(editSpinnerDay.getSelectedItem().toString());  
 course.setPriceYoga(editPriceYoga.getText().toString());  
 course.setTimeYoga(editSpinnerTime.getSelectedItem().toString());  
 course.setCapacityYoga(editCapacityYoga.getText().toString());  
 course.setDurationYoga(editDurationYoga.getText().toString());  
 course.setDescriptionYoga(editDescriptionYoga.getText().toString());  
  
 String priceYoga = editPriceYoga.getText().toString();  
 String capacityYoga = editCapacityYoga.getText().toString();  
 String durationYoga = editDurationYoga.getText().toString();  
 String descriptionYoga = editDescriptionYoga.getText().toString();  
  
 boolean check = validateInfo(priceYoga,  
 capacityYoga, durationYoga, descriptionYoga);  
  
 if (check) {  
 db.updateCourse(course);  
 notifyItemChanged(getAdapterPosition(), course);  
 }  
 else {  
 Toast.*makeText*(context, "Sorry, Check the Information", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 dialog.dismiss();  
 }  
  
 private Boolean validateInfo(String priceYoga, String capacityYoga, String durationYoga, String descriptionYoga)  
 {  
 if (priceYoga.length() == 0)  
 {  
 editPriceYoga.requestFocus();  
 editPriceYoga.setError("Please enter price");  
 return false;  
 }  
 else if (capacityYoga.length() == 0) {  
 editCapacityYoga.requestFocus();  
 editCapacityYoga.setError("Please enter capacity");  
 return false;  
 }  
 else if (durationYoga.length() == 0) {  
 editDurationYoga.requestFocus();  
 editDurationYoga.setError("Please enter duration");  
 return false;  
 }  
 else if (descriptionYoga.length() == 0) {  
 editDescriptionYoga.requestFocus();  
 editDescriptionYoga.setError("Please enter description");  
 return false;  
 }  
 return true;  
 }  
  
 });  
 }  
  
 public void deleteButton(int id)  
 {  
 alertDialogBuilder = new AlertDialog.Builder(context);  
  
 inflater = LayoutInflater.*from*(context);  
 View view = inflater.inflate(R.layout.*confirm\_dialog*, null);  
  
 Button noButton = (Button) view.findViewById(R.id.*noButton*);  
 Button yesButton = (Button) view.findViewById(R.id.*yesButton*);  
  
 alertDialogBuilder.setView(view);  
 dialog = alertDialogBuilder.create();  
 dialog.show();  
  
 noButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 dialog.dismiss();  
 }  
 });  
  
 yesButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 CourseDatabaseHandle db = new CourseDatabaseHandle(context);  
 db.deleteCourse(id);  
 courseList.remove(getAdapterPosition());  
 notifyItemRemoved(getAdapterPosition());  
  
 dialog.dismiss();  
  
 }  
 });  
 }  
  
 public void changeClassActivity(int id)  
 {  
 Intent intent = new Intent(context, ClassInstanceActivity.class);  
 intent.putExtra("course\_id", id);  
 context.startActivity(intent);  
 }  
  
 }  
  
}

ClassRecyclerViewAdapter.java

public class ClassRecyclerViewAdapter extends RecyclerView.Adapter<ClassRecyclerViewAdapter.ViewHolder> {  
 private Context context;  
 private List<ClassYoga> classYogaList;  
 private LayoutInflater inflater;  
  
 private AlertDialog.Builder alertDialogBuilder;  
 private AlertDialog dialog;  
  
 public ClassRecyclerViewAdapter(Context context, List<ClassYoga> classYogaList) {  
 this.context = context;  
 this.classYogaList = classYogaList;  
 }  
  
 @NonNull  
 @Override  
 public ClassRecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.*class\_list*, parent, false);  
 return new ClassRecyclerViewAdapter.ViewHolder(view, context);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ClassRecyclerViewAdapter.ViewHolder holder, int position) {  
 ClassYoga classYoga = classYogaList.get(position);  
  
 holder.dateClass.setText(classYoga.getClassDate());  
 holder.teacherClass.setText(classYoga.getTeacherName());  
 holder.commentClass.setText(classYoga.getComment());  
  
 }  
  
 @Override  
 public int getItemCount() {  
 return classYogaList.size();  
 }  
  
 public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
 private TextView dateClass;  
 private TextView teacherClass;  
 private TextView commentClass;  
  
 private Button editButtonClass;  
 private Button deleteButtonClass;  
  
  
 public ViewHolder(@NonNull View view, Context ctx) {  
 super(view);  
 context = ctx;  
  
 dateClass = (TextView) view.findViewById(R.id.*dateClass*);  
 teacherClass = (TextView) view.findViewById(R.id.*teacherClass*);  
 commentClass = (TextView) view.findViewById(R.id.*commentClass*);  
  
  
 editButtonClass = (Button) view.findViewById(R.id.*editButtonClass*);  
 deleteButtonClass = (Button) view.findViewById(R.id.*deleteButtonClass*);  
  
 editButtonClass.setOnClickListener(this);  
 deleteButtonClass.setOnClickListener(this);  
  
 view.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 // Go to next screen => DetailsClassYogaActivity  
// int position = getAdapterPosition();  
// ClassYoga classYoga = classYogaList.get(position);  
// Intent intent = new Intent(context, DetailCourseActivity.class);  
//  
// intent.putExtra("detailTypeYogaInClass", classYoga.getCourse\_id());  
// intent.putExtra("detailDateClass", classYoga.getClassDate());  
// intent.putExtra("detailTeacherClass", classYoga.getTeacherName());  
//  
// context.startActivity(intent);  
 }  
 });  
 }  
  
 @Override  
 public void onClick(View v) {  
 int position = getAdapterPosition();  
 ClassYoga classYoga = classYogaList.get(position);  
  
 if (v.getId() == R.id.*editButtonClass*) {  
 editButtonClass(classYoga);  
 }  
  
 if (v.getId() == R.id.*deleteButtonClass*) {  
 deleteButtonClass(classYoga.getId());  
 }  
 }  
  
 public void editButtonClass(ClassYoga classYoga) {  
 alertDialogBuilder = new AlertDialog.Builder(context);  
  
 inflater = LayoutInflater.*from*(context);  
 View view = inflater.inflate(R.layout.*edit\_class*, null);  
  
 final TextView title = (TextView) view.findViewById(R.id.*titleYogaClass*);  
 final TextView editDateClass = (TextView) view.findViewById(R.id.*editDateClass*);  
 final EditText editTeacherClass = (EditText) view.findViewById(R.id.*editTeacherClass*);  
 final EditText editCommentClass = (EditText) view.findViewById(R.id.*editCommentClass*);  
  
 title.setText("Edit Class Yoga");  
 final Calendar calendar = Calendar.*getInstance*();  
 final int year = calendar.get(Calendar.*YEAR*);  
 final int month = calendar.get(Calendar.*MONTH*);  
 final int day = calendar.get(Calendar.*DAY\_OF\_MONTH*);  
  
 int course\_id = classYoga.getCourse\_id();  
 editDateClass.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 DatePickerDialog datePickerDialog = new DatePickerDialog(context, new DatePickerDialog.OnDateSetListener() {  
 @Override  
 public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {  
 CourseDatabaseHandle courseDB = new CourseDatabaseHandle(context);  
 if (courseDB.getCourse(course\_id) != null)  
 {  
 Course course = courseDB.getCourse(course\_id);  
 String dayCourse = course.getDayYoga();  
 String day\_of\_the\_week\_class = DateUtil.*showDayOfTheWeek*(dayOfMonth, month + 1, year);  
 if(!day\_of\_the\_week\_class.equals(dayCourse)) {  
 Toast.*makeText*(context, "The date does not match the day of the week!", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 @SuppressLint("DefaultLocale") String date = String.*format*("%02d/%02d/%d", dayOfMonth, month + 1, year);  
 Toast.*makeText*(context, "Select date: " + date, Toast.*LENGTH\_SHORT*).show();  
 editDateClass.setText(date);  
 }  
 }  
 }, year, month, day);  
 datePickerDialog.show();  
 }  
 });  
  
 editTeacherClass.setText(classYoga.getTeacherName());  
 editCommentClass.setText(classYoga.getComment());  
  
 Button saveButtonClass = (Button) view.findViewById(R.id.*saveEditButtonClass*);  
  
  
 alertDialogBuilder.setView(view);  
 dialog = alertDialogBuilder.create();  
 dialog.show();  
  
 saveButtonClass.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 ClassDatabaseHandle db = new ClassDatabaseHandle(context);  
  
 // Update class  
 classYoga.setClassDate(editDateClass.getText().toString().trim());  
 classYoga.setTeacherName(editTeacherClass.getText().toString());  
 classYoga.setComment(editCommentClass.getText().toString());  
  
  
 String dateClass = editDateClass.getText().toString().trim();  
 String teacherClass = editTeacherClass.getText().toString();  
  
 boolean check = validateInfo(dateClass, teacherClass);  
  
 if (check) {  
 db.updateClassYoga(classYoga);  
 notifyItemChanged(getAdapterPosition(), classYoga);  
 } else {  
 Toast.*makeText*(context, "Sorry, Check the Information Class", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 dialog.dismiss();  
 }  
  
 private Boolean validateInfo(String dateClass, String teacherClass) {  
 if (dateClass.length() == 0) {  
 editDateClass.requestFocus();  
 editDateClass.setError("Please enter price");  
 return false;  
 } else if (teacherClass.length() == 0) {  
 editTeacherClass.requestFocus();  
 editTeacherClass.setError("Please enter capacity");  
 return false;  
  
 }  
 return true;  
 }  
 });  
 }  
 public void deleteButtonClass(int id) {  
 alertDialogBuilder = new AlertDialog.Builder(context);  
  
 inflater = LayoutInflater.*from*(context);  
 View view = inflater.inflate(R.layout.*confirm\_dialog*, null);  
  
 Button noButton = (Button) view.findViewById(R.id.*noButton*);  
 Button yesButton = (Button) view.findViewById(R.id.*yesButton*);  
  
 alertDialogBuilder.setView(view);  
 dialog = alertDialogBuilder.create();  
 dialog.show();  
  
 noButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 dialog.dismiss();  
 }  
 });  
  
 yesButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 ClassDatabaseHandle db = new ClassDatabaseHandle(context);  
 db.deleteClassYoga(id);  
 classYogaList.remove(getAdapterPosition());  
 notifyItemRemoved(getAdapterPosition());  
  
 dialog.dismiss();  
  
 }  
 });  
 }  
 }  
}

Constants.java

public class Constants  
{  
 // Yoga Course DB  
 public static final int *DB\_VERSION* = 1;  
 public static final String *COURSE\_DB\_NAME* = "YogaAppDB";  
 public static final String *COURSE\_TABLE\_NAME* = "YogaCourseTBL";  
  
 // Table columns Yoga Course DB  
 public static final String *COURSE\_KEY\_ID* = "id";  
 public static final String *COURSE\_KEY\_TYPE\_YOGA* = "type\_yoga";  
 public static final String *COURSE\_KEY\_DAY\_YOGA* = "day\_yoga";  
 public static final String *COURSE\_KEY\_PRICE\_YOGA* = "price\_yoga";  
 public static final String *COURSE\_KEY\_TIME\_YOGA* = "time\_yoga";  
 public static final String *COURSE\_KEY\_CAPACITY\_YOGA* = "capacity\_yoga";  
 public static final String *COURSE\_KEY\_DURATION\_YOGA* = "duration\_yoga";  
 public static final String *COURSE\_KEY\_DESCRIPTION\_YOGA* = "description\_yoga";  
  
  
// // Class Yoga Course DB  
 public static final String *CLASS\_INSTANCE\_DB\_NAME* = "ClassYogaDB";  
 public static final String *CLASS\_INSTANCE\_TABLE\_NAME* = "ClassInstanceTBL";  
//  
// // Table columns Class DB  
 public static final String *CLASS\_INSTANCE\_KEY\_ID* = "id";  
 public static final String *CLASS\_INSTANCE\_KEY\_COURSE\_ID* = "course\_id";  
 public static final String *CLASS\_INSTANCE\_KEY\_TEACHER\_NAME* = "teacher\_name";  
 public static final String *CLASS\_INSTANCE\_KEY\_CLASS\_DATE* = "class\_date";  
 public static final String *CLASS\_INSTANCE\_KEY\_COMMENT* = "comments";  
  
  
}

DateUtil.java

public class DateUtil {  
  
 public static String showDayOfTheWeek(int day, int month, int year) {  
 Calendar calendar = Calendar.*getInstance*();  
 calendar.set(year, month - 1, day);  
 int dayOfWeek = calendar.get(Calendar.*DAY\_OF\_WEEK*);  
 switch (dayOfWeek) {  
 case Calendar.*SUNDAY*:  
 return "Sunday";  
 case Calendar.*MONDAY*:  
 return "Monday";  
 case Calendar.*TUESDAY*:  
 return "Tuesday";  
 case Calendar.*WEDNESDAY*:  
 return "Wednesday";  
 case Calendar.*THURSDAY*:  
 return "Thursday";  
 case Calendar.*FRIDAY*:  
 return "Friday";  
 case Calendar.*SATURDAY*:  
 return "Saturday";  
 default:  
 return "Unknown";  
 }  
 }  
}