# MỤC LỤC

[CHƯƠNG 1. MỞ ĐẦU 2](#_Toc469991409)

[1. Giới thiệu môn học và nhóm thực hiện 2](#_Toc469991410)

[2. Mô tả ứng dụng 2](#_Toc469991411)

[3. Sự cần thiết của ứng dụng 3](#_Toc469991412)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 4](#_Toc469991413)

[2.1 Phân tích hệ thống 4](#_Toc469991414)

[2.2 Thiết kế hệ thống 10](#_Toc469991419)

[CHƯƠNG 3. CÀI ĐẶT VÀ KIỂM THỬ 13](#_Toc469991420)

[3.1 Cài đặt 13](#_Toc469991421)

[3.2 Kiểm thử 40](#_Toc469991426)

[CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC 40](#_Toc469991427)

[4.1 Kết quả đạt được 41](#_Toc469991428)

[4.2 Các kết luận và kiến nghị 46](#_Toc469991429)

[TÀI LIỆU THAM KHẢO 48](#_Toc469991430)

# CHƯƠNG 1. MỞ ĐẦU

## 1. Giới thiệu môn học và nhóm thực hiện

* Giới thiệu môn học:
* Android đang là hệ điều hành chiến lĩnh thị trường thiết bị động hiện nay. Hàng trăm ngàn thiết bị từ tivi, máy nghe nhạc,... cho đến điện thoại di động đều có thể chạy trên nền Android.
* Với xu hướng ngày càng phát triển của Android, việc lập trình Android chính là cơ hội tuyệt vời nhất cho các sinh viên ra trường đủ khả năng ứng tuyển vị trí lập trình Android tại các doanh nghiệp phần mềm.
* Giới thiệu nhóm thực hiện:
* Nhóm Android-2 có 7 thành viên:

1. Phạm Thanh Huy (Nhóm trưởng)
2. Võ Thành Nhơn
3. Đinh Đình Hiệp
4. Phạm Chí Hải
5. Đỗ Hữu Thịnh
6. Đinh Chí Khang
7. Lê Cao Thế.

* Nhóm Android-2 được thành lập ngày 07/11/2016, bởi 7 thành viên trên của lớp CD14TT2 tại Trường Cao đẳng Công nghê Thủ Đức.
* Mục đích thành lập nhóm: nhằm chia sẽ kinh nghiệm học tập để hoàn thành đồ án cuối kỳ của môn Lập trình Android nâng cao.

## 2. Mô tả ứng dụng

* RSS là một định dạng tập tin thuộc họ XML dùng trong việc chia sẻ tin tức web.
* RSS (Viết tắt từ Really Simple Syndication) là một tiêu chuẩn định dạng tài liệu dựa trên XML nhằm giúp người sử dụng dễ dàng cập nhật và tra cứu thông tin một cách nhanh chóng và thuận tiện nhất bằng cách tóm lược thông tin vào trong một đoạn dữ liệu ngắn gọn, hợp chuẩn.
* Dữ liệu này được các chương trình đọc tin chuyên biệt (gọi là News reader) phân tích và hiển thị trên máy tính của người sử dụng. Trên trình đọc tin này, người sử dụng có thể thấy những tin chính mới nhất, tiêu đề tóm tắt và cả đường link để xem toàn bộ tin.

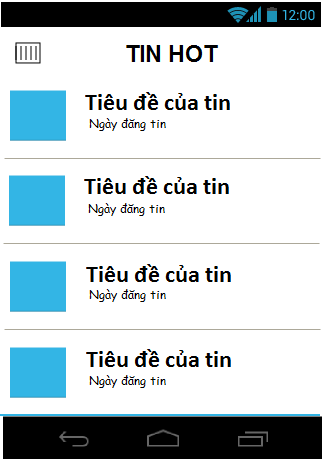
## 3. Sự cần thiết của ứng dụng

* **RSS** có thể tái sử dụng. Thông tin trong **RSS** hoàn toàn có thể tái sử dụng cho nhiều mục đích: cung cấp cho các kênh thông tin khác (nằm trong feed khác), tích hợp vào các thư viện động hoặc nguyên liệu phục vụ quá trình học tập.
* **RSS** an toàn. Xuất phát từ phương thức truyền tải, **RSS** không thể mang theo virus hoặc trojan như newsletter hoặc các tệp tin đính kèm
* **RSS** linh hoạt trong cách cung cấp thông tin. Người sử dụng tiết kiệm được thời gian, thông tin được cung cấp ở dạng tóm tắt (thường chỉ là các tiêu đề- title).

# CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 2.1 Phân tích hệ thống

* Chức năng ở Màn hình Mobile chính của tin
* Giao diện người dùng

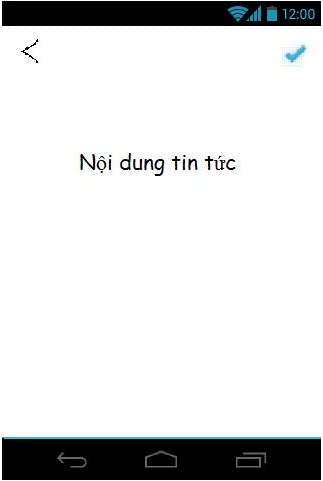


Hình 1: Màn hình chính

* Yêu cầu chức năng.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| **Màn hình Mobile Trang chủ tin** | Màn hình mobile trang chủ tin có:   * 1 icon (Menu) * 1 Text View (Tiêu đề của tin) * 1 List View:   + 1 Image View  + 2 Text View (Nội dung tin, Ngày đăng) | N/A | N/A |
| **Icon Menu** | Sử dụng để mở danh sách các chuyên mục của tin tức. | Bấm vào icon Menu | Hiển thị Slidebar Menu cho người dùng chọn danh mục |
| **ListView** | Hiển thị các tin tức của một chuyên mục nào đó. | N/A | N/A |
| **Item ListView** | Chọn một item trong ListView để đọc nội dung chi tiết của tin đó | Bấm vào bất kỳ 1 item trong listview | Đi đến màn hình chứa đầy đủ thông tin chi tiết của item vừa chọn |

* Chức năng ở Màn hình Mobile chi tiết của tin
* Giao diện người dùng



Hình 2: Màn hình chi tiết

* Yêu cầu chức năng:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu đề** | **Mô tả** | **Quy trình** | **Kết quả** |
| Màn hình Mobile Chi tiết | Màn hình Mobile Chi tiết có 1 button, 1 check box và 1 webview | N/A | N/A |
| Icon Quay lại | Sử dụng để quay lại màn hình chính | Bấm vào nút button quay lại | Quay lại màn hình chính |
| Checkbox | Sử dụng để đánh dấu tin đã được đọc sẻ lưu vào database | Bấm vào checkbox | Check thành công |
| WebView | Sử dụng để hiện thông tin chi tiết của tin đã được chọn | N/A | N/A |

* Chức năng ở Màn hình Mobile chi tiết của tin
* Giao diện người dùng



Hình 3: Màn hình SlideBarMenu

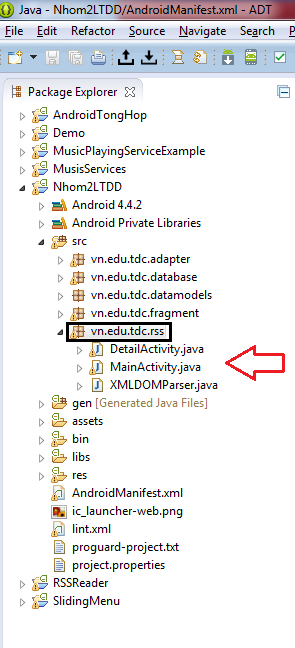
* Yêu cầu chức năng:

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Description** | **Action** | **Response** |
| **Màn hình Mobile Slide Menu** | Màn hình slide menu có 1 list view   * Trong list view có 8 danh mục:   + Thể thao  + Giáo dục  + Kinh doanh  + Du lịch  + Thế giới  + Giải trí  + Sức khỏe  + Pháp luật   * Mỗi danh mục tương ứng có mỗi icon đại diện | N/A | N/A |
| **Danh mục thể thao** | Hiển thị chủ đề thể thao | Bấm vào danh mục thể thao | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề thể thao |
| **Danh mục giáo dục** | Hiển thị chủ đề giáo dục | Bấm vào danh mục giáo dục | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề giáo dục |
| **Danh mục kinh doan** | Hiển thị chủ đề kinh doanh | Bấm vào danh mục kinh doanh | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề kinh doanh |
| **Danh mục du lịch** | Hiển thị chủ đề du lịch | Bấm vào danh mục du lịch | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề du lịch |
| **Danh mục thế giới** | Hiển thị chủ đề thế giới | Bấm vào danh mục thế giới | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề thế giới |
| **Danh mục giải trí** | Hiển thị chủ đề giải trí | Bấm vào danh mục giải trí | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề giải trí |
| **Danh mục sức khỏe** | Hiển thị chủ đề sức khỏe | Bấm vào danh mục sức khỏe | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề sức khỏe |
| **Danh mục pháp luật** | Hiển thị chủ đề pháp luật | Bấm vào danh mục pháp luật | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề pháp luật |
| **Danh mục lịch sử** | Hiển thị chủ đề lịch sử chứa các tin đã lưu từ database | Bấm vào danh mục lịch sử | Slide Menu đóng lại và hiện lên màn hình chứa các tin tức của chủ đề lịch sử |
| **Icon Menu** | Sử dụng để mở từng danh mục chủ đề của tin tức | Bấm vào Button Menu | Slide Menu đóng lại |

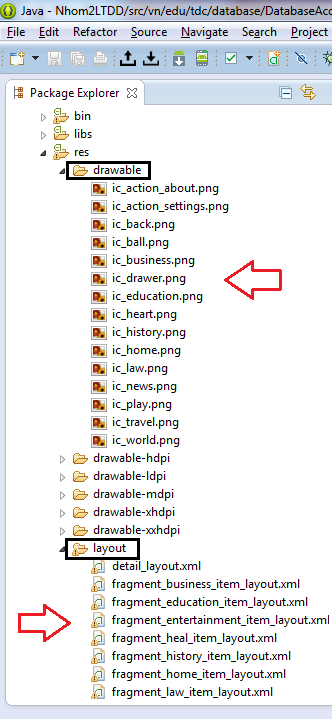
## 2.2 Thiết kế hệ thống

**2.2.1 Mô tả chương trình**

* Môi trường hoạt động: Trên eclipse thư mục code android được bố trí như sau:



* Các package chứa code sẽ nằm trong thư mục src:
* Cụ thể là: Package vn.edu.tdc.rss chứa các class và Activity của project.
* Các tài nguyên hệ thống như ảnh, layout, string, dùng chung sẽ được tổ chức trong thư mục res.



* Một file khai báo rất quan trọng để gắn kết các thành phần trong toàn bộ hệ thống đó là file AndroidManifest.xml

# CHƯƠNG 3. CÀI ĐẶT VÀ KIỂM THỬ

## Cài đặt

**3.1.1. Màn hình chính: MainActivity.java**

* Giao diện gồm có: 1 icon menu trên thanh Action Bar, 1 Fragment hiển thị danh sách các bài viết, 1 tiêu đề chính.
* Cách xây dựng SlideBarMenu bằng cách dùng thủ thuật Navigation Drawer trong android.
* Bước 1: Xây dựng giao diện của **main\_layout.xml**

<android.support.v4.widget.DrawerLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:id=*"@+id/drawer\_layout"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:orientation=*"horizontal"* >

<!-- The main content view -->

<LinearLayout

android:id=*"@+id/newsContent"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:orientation=*"vertical"* >

</LinearLayout>

<!-- The navigation drawer -->

<ListView

android:id=*"@+id/slider\_list"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:layout\_gravity=*"start"*

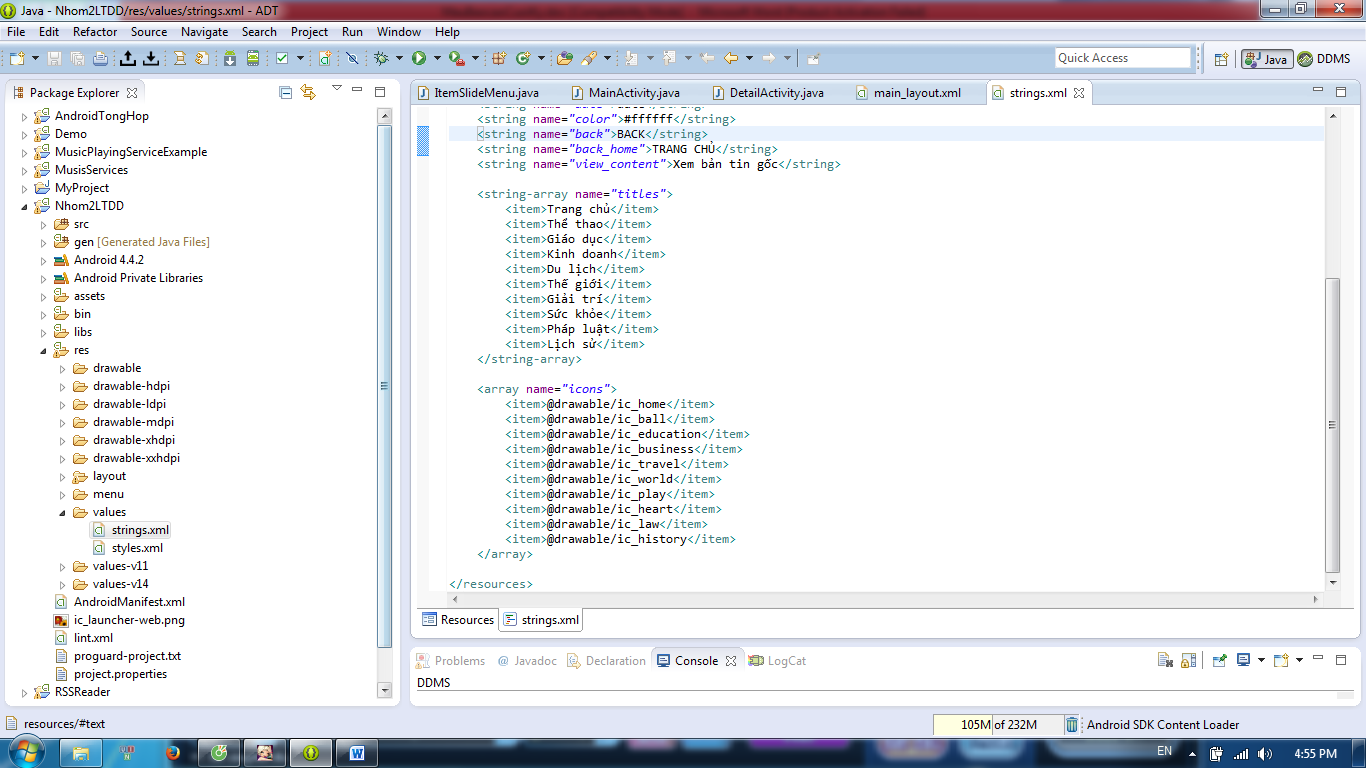
android:background=*"#ffffff"*

android:choiceMode=*"singleChoice"*

android:dividerHeight=*"0dp"* />

</android.support.v4.widget.DrawerLayout>

* Bước 2: Tạo text và icon: Tại mục res/values trong file **string.xml**



* Bước 3: Custom ListView: Cần 1 tập tin bố trí các item của listview để có thể tùy chỉnh việc hiển thị theo ý muốn. Vào res/layout và tạo 1 file có tên **listview\_item\_layout.xml** như sau:

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"horizontal"* >

<ImageView

android:id=*"@+id/imgItems"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:background=*"@drawable/ic\_launcher"* />

<LinearLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center\_vertical"*

android:layout\_marginLeft=*"15dp"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/txtTitle"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/title"*

android:textSize=*"18sp"*

android:textStyle=*"bold"* />

<TextView

android:id=*"@+id/txtDate"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:text=*"@string/date"*

android:textSize=*"10sp"*

android:textStyle=*"italic"* />

</LinearLayout>

</LinearLayout>

## Theo như trên thì 1 item sẽ có 2 phần đó là icon và 1 phần Text chứa nội dung hiển thị

## Bước 4: Tạo mới 1 models có tên là ItemSlideMenu.java có nội dung như sau:

**public** **class** ItemSlideMenu {

**private** **int** icon;

**private** String title;

**public** **int** getIcon() {

**return** icon;

}

**public** **void** setIcon(**int** icon) {

**this**.icon = icon;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** ItemSlideMenu(**int** icon, String title) {

**super**();

**this**.icon = icon;

**this**.title = title;

}

}

## Bước 5: Để tiếp tục việc tùy chỉnh listview trong Navigation Drawer thì ta cần tạo một lớp có tên là SlideMenuAdapter.java

**public** **class** SlideMenuAdapter **extends** ArrayAdapter<ItemSlideMenu> {

Activity context;

ArrayList<ItemSlideMenu> arrItems;

**public** SlideMenuAdapter(Activity context, ArrayList<ItemSlideMenu> arrItems) {

**super**(context, -1, arrItems);

**this**.context = context;

**this**.arrItems = arrItems;

}

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

**if** (convertView == **null**) {

LayoutInflater inflater = context.getLayoutInflater();

convertView = inflater.inflate(R.layout.*slide\_menu\_item\_layout*, **null**);

}

ImageView imgIcon = (ImageView) convertView.findViewById(R.id.*icon*);

TextView txtTitle = (TextView) convertView.findViewById(R.id.*title*);

ItemSlideMenu items = getItem(position);

**if** (items != **null**) {

// Anh xa + Gan gia tri

imgIcon.setImageResource(items.getIcon());

txtTitle.setText(items.getTitle());

}

**return** convertView;

}

}

* Bước 6: Thao tác với Navigation Drawer
* Đến bước này thì ta đã cơ bản hoàn thành việc tùy chỉnh cho list các item trong thanh Navigation Drawer. Bây giờ chúng ta sẽ chuyển sang làm việc với MainActivity.java để có thể thực hiện các thao tác với Navigation Drawer
* Bây giờ ta thêm dữ liệu vào list Menu

**for** (**int** i = 0; i < menutitles.length; i++) {

ItemSlideMenu items = **new** ItemSlideMenu(menuIcons.getResourceId(i,-1), menutitles[i]);

itemSlide.add(items);

}

* Gán vào Adapter:

menuadapter = **new** SlideMenuAdapter(MainActivity.**this**, itemSlide);

lvDrawerMenu.setAdapter(menuadapter);

* Tiếp tục ta bắt sự kiện khi click vào menu của list trong Navigation Drawer. Ở đây em chỉ bắt sự kiện khi người dùng click vào nút Trang chủ của menu bên trái thì sẽ chuyển hướng sang Fragment. Còn các sự kiện của các menu khác thì làm tương tự.
* Tạo mới 1 Fragment có tên là **HomeContent.java**

**package** vn.edu.tdc.fragment;

**import** android.app.Activity;

**import** android.os.Bundle;

**public** **class** HomeContent **extends** Fragment{

@Override

**public** View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// **TODO** Auto-generated method stub

**return** **super**.onCreateView(inflater, container, savedInstanceState);

}

}

* Tạo mới 1 giao diện có tên là **fragmem\_home\_item\_layout.xml**

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:orientation=*"vertical"* >

<LinearLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"0dp"*

android:layout\_weight=*"1"*

android:gravity=*"center"*

android:orientation=*"horizontal"* >

<TextView

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:gravity=*"center"*

android:text=*"@string/home"*

android:textSize=*"25sp"*

android:textStyle=*"bold"* />

</LinearLayout>

<LinearLayout

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"0dp"*

android:layout\_weight=*"9"*

android:orientation=*"horizontal"* >

<ListView

android:id=*"@+id/lvHome"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"* />

</LinearLayout>

</LinearLayout>

* Bây giờ ta bắt sự kiện khi click vào menu của list trong Navigation Drawer.

**class** SlideitemListener **implements** ListView.OnItemClickListener {

@Override

**public** **void** onItemClick(AdapterView<?> parent, View view, **int** position,

**long id) {**

updateDisplay(position);

}

}

**private** **void** updateDisplay(**int** position) {

Fragment fragment = **null**;

**switch** (position) {

**case** 0:

fragment = **new** HomeContent();

**break**;

**default**:

**break**;

}

**if** (fragment != **null**) {

FragmentManager fragmentManager = getFragmentManager();

fragmentManager.beginTransaction()

.replace(R.id.*newsContent*, fragment).commit();

// update selected item and title, then close the drawer

setTitle(menutitles[position]);

drawerLayout.closeDrawer(lvDrawerMenu);

} **else** {

// error in creating fragment

Log.*e*("MainActivity", "Error in creating fragment");

}

}

* Muốn đọc được nội dung của RSS. Việc đầu tiên ta sẽ sử dụng 1 class ReadData, chúng ta sẽ sử dụng hàm AsyncTask là hàm đa phương thức dùng để đọc và gữi dữ liệu đi

**class** ReadData **extends** AsyncTask<String, Integer, String> {

@Override

**protected** **void** onPreExecute() {

// **TODO** Auto-generated method stub

**super**.onPreExecute()

}

@Override

**protected** String doInBackground(String... arg0) {

// **TODO** Auto-generated method stub

**return** *null;*

}

@Override

**protected** **void** onPostExecute(String result) {

**super**.onPostExecute(result);

}

}

* Để có thể đọc được một đường dẫn URL từ Internet, ta có đoạn code sau:

**private** **static** String readContentFromUrl(String strUrl) {

StringBuilder content = **new** StringBuilder();

**try** {

// create a url object

URL url = **new** URL(strUrl);

// create a urlconnection object

URLConnection urlConnection = url.openConnection();

// wrap the urlconnection in a bufferedreader

BufferedReader bufferedReader = **new** BufferedReader(

**new** InputStreamReader(urlConnection.getInputStream()));

String line;

// read from the urlconnection via the bufferedreader

**while** ((line = bufferedReader.readLine()) != **null**) {

content.append(line + "\n");

}

bufferedReader.close();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** content.toString();

}

* Để có thể thực thi class ReadData ta vào onCreateView khai báo đoạn code sau:

Runnable runnable = **new** Runnable() {

@Override

**public** **void** run() {

// **TODO** Auto-generated method stub

**new** ReadData().execute("Ten duong dan url");

}

};

runnable.run();

* Vì chúng ta lấy dữ liệu về từ Internet nên ta phải khai báo quyền Internet trong file **AndroidManifest.xml**

<use-permission android:name=*”android.permission.INTERNET”* />

* Để đọc được XML từ một đường dẫn URL ta có 2 bước:
* Bước 1: Tạo Class **XMLDOMParser.java**

**public** **class** XMLDOMParser {

**public** Document getDocument(String xml)

{

Document document = **null**;

DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();

**try**{

DocumentBuilder db = factory.newDocumentBuilder();

InputSource is = **new** InputSource();

is.setCharacterStream(**new** StringReader(xml));

is.setEncoding("UTF-8");

document = db.parse(is);

}**catch**(ParserConfigurationException e)

{

Log.*e*("Error: ", e.getMessage(), e);

**return** **null**;

}

**catch** (SAXException e) {

Log.*e*("Error: ", e.getMessage(), e);

**return** **null**;

}

**catch**(IOException e){

Log.*e*("Error: ", e.getMessage(), e);

**return** **null**;

}

**return** document;

}

**public** String getValue(Element item, String name)

{

NodeList nodes = item.getElementsByTagName(name);

**return** **this**.getTextNodeValue(nodes.item(0));

}

**private** **final** String getTextNodeValue(Node elem) {

Node child;

**if**( elem != **null**){

**if** (elem.hasChildNodes()){

**for**( child = elem.getFirstChild(); child != **null**; child = child.getNextSibling() ){

**if**( child.getNodeType() == Node.*TEXT\_NODE* ){

**return** child.getNodeValue();

}

}

}

}

**return** "";

}

}

* Bước 2: Vào **HomeContent.java** gọi lại class XMLDOMParser trong hàm onPostExcute của class ReadData. Trong XMLDOMParser có những hàm dùng để tương tác với HTML.

**class** ReadData **extends** AsyncTask<String, Integer, String> {

@Override

**protected** **void** onPreExecute() {

// **TODO** Auto-generated method stub

**super**.onPreExecute();

}

@Override

**protected** String doInBackground(String... arg0) {

// **TODO** Auto-generated method stub

**return** *null;*

}

@Override

**protected** **void** onPostExecute(String result) {

// **TODO** Auto-generated method stub

XMLDOMParser parser = **new** XMLDOMParser();

Document document = parser.getDocument(result);

NodeList nodeList = document.getElementsByTagName("item");

NodeList nodeListDescription = document

.getElementsByTagName("description");

String photo = "";

String title = "";

String link = "";

String date = "";

String description = "";

**for** (**int** i = 0; i < nodeList.getLength(); i++) {

String cdata = nodeListDescription.item(i + 1).getTextContent();

Pattern pattern = Pattern

.*compile*("<img[^>]+src\\s\*=\\s\*['\"]([^'\"]+)['\"][^>]\*>");

Matcher matcher = pattern.matcher(cdata);

**if** (matcher.find()) {

photo = matcher.group(1);

}

Element element = (Element) nodeList.item(i);

title = parser.getValue(element, "title");

date = parser.getValue(element, "pubDate");

description = parser.getValue(element, "description");

link = parser.getValue(element, "link");

description));

arrNews.add(**new** News(title, photo, date, link, description));

}

adapter = **new** NewsAdapter(getActivity(),arrNews);

lventertainment.setAdapter(adapter);

**super**.onPostExecute(result);

}

}

* Tạo data models class News.java cho bài viết: dùng để truyền các tham số vào mảng

**public** **class** News {

**private** **int** id;

**private** String strTitle;

**private** String strImage;

**private** String strDate;

**private** String strLink;

**private** String strDesctiption;

**public** News() {

}

**public** News(String strTitle, String strImage, String strDate,

String strLink, String strDesctiption) {

**super**();

**this**.strTitle = strTitle;

**this**.strImage = strImage;

**this**.strDate = strDate;

**this**.strLink = strLink;

**this**.strDesctiption = strDesctiption;

}

**public** News(String strTitle, String strImage, String strDate, String strLink) {

**super**();

**this**.strTitle = strTitle;

**this**.strImage = strImage;

**this**.strDate = strDate;

**this**.strLink = strLink;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getStrTitle() {

**return** strTitle;

}

**public** **void** setStrTitle(String strTitle) {

**this**.strTitle = strTitle;

}

**public** String getStrImage() {

**return** strImage;

}

**public** **void** setStrImage(String strImage) {

**this**.strImage = strImage;

}

**public** String getStrDate() {

**return** strDate;

}

**public** **void** setStrDate(String strDate) {

**this**.strDate = strDate;

}

**public** String getStrLink() {

**return** strLink;

}

**public** **void** setStrLink(String strLink) {

**this**.strLink = strLink;

}

**public** String getStrDesctiption() {

**return** strDesctiption;

}

**public** **void** setStrDesctiption(String strDesctiption) {

**this**.strDesctiption = strDesctiption;

}

}

* Tạo class **NewsAdapter.java**

**public** **class** NewsAdapter **extends** ArrayAdapter<News> {

Activity context = **null**;

ArrayList<News> arrNews = **null**;

**public** NewsAdapter(Activity context, ArrayList<News> arrNews) {

**super**(context, -1, arrNews);

**this**.context = context;

**this**.arrNews = arrNews;

}

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

**if** (convertView == **null**) {

LayoutInflater inflater = context.getLayoutInflater();

convertView = inflater.inflate(R.layout.*listview\_item\_layout*, **null**);

}

ImageView imgItems = (ImageView) convertView.findViewById(R.id.*imgItems*);

TextView txtTitle = (TextView) convertView.findViewById(R.id.*txtTitle*);

TextView txtDate = (TextView) convertView.findViewById(R.id.*txtDate*);

News news = getItem(position);

**if** (news != **null**) {

// Anh xa + Gan gia tri

txtTitle.setText(news.getStrTitle());

txtDate.setText(news.getStrDate());

Picasso.*with*(getContext()).load(news.getStrImage()).into(imgItems);

}

**return** convertView;

}

}

* Cuối cùng về **HomeContent.java** ta sẽ set Adapter cho bài viết. Đã làm ở bước2.

***3.1.2. Màn hình chi tiết (DetailActivity.java)***

* Xét sự kiện cho Listview ở class **HomeContent.java**. Gọi biến intent để truyền dữ liệu đi.

lvHome.setOnItemClickListener(**new** OnItemClickListener() {

@Override

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2,

**long** arg3) {

// **TODO** Auto-generated method stub

Bundle bundle = **new** Bundle();

bundle.putString("title", arrNews.get(arg2).getStrTitle());

bundle.putString("image", arrNews.get(arg2).getStrImage());

bundle.putString("date", arrNews.get(arg2).getStrDate());

bundle.putString("link", arrNews.get(arg2).getStrLink());

Intent intent = **new** Intent(getActivity(), DetailActivity.**class**);

intent.putExtra("news", bundle);

startActivity(intent);

}

});

* Về phía DetailActivity.java ta gọi lại biến intent để nhận dữ liệu về

**final** Intent intent = getIntent();

f**inal** Bundle bundle = intent.getBundleExtra("news");

link = bundle.getString("link");

title = bundle.getString("title");

image = bundle.getString("image");

date = bundle.getString("date");

* Để đọc được nội dung chi tiết của bài viết đó. Ta sử dụng thư viện Jsoup để đọc mã nguồn HTML ta có 3bước
* Bước 1: Ta lấy tất cả các thứ cần trong 1 bài viết, là title, date, description và nội dung chính từ Jsoup. Ta đọc mã nguồn và đi vào sâu trong nó để lấy các giá trị ra.

Document doc = Jsoup.*connect*(link).get();

Elements pubDate = doc.select("div.block\_timer\_share div.block\_timer");

Elements title = doc.select("div.title\_news h1");

Elements description = doc.select("h3.short\_intro");

Elements content = doc.select("div.fck\_detail");

* Bước 2: Phân tích HTML trong Android.

detail += "<font size=\" 1.2em \" style = \" color: #005500 \"><em>"

+ pubDate.text() + "</em></font>";

detail += "<h2 style = \" color: red \">" + title.text()

+ "</h2>";

detail += "<h3>" + description.text() + "</h3>";

detail += "<p>" + content.text() + "</p>";

* Bước 3: Ta dùng webview để load đoạn mã trên.

Webview.loadDataWithBaseURL("”, detail, “text/html”, “UTF-8”, “”);

## Tạo Database cho các tin tức lưu.

* Bước 1: Tạo 1 lớp **DatabaseHelper.java** kế thừa từ **SQLiteOpenHelper**.

**public** **class** DatabaseHelper **extends** SQLiteOpenHelper {

**static** **int** *DB\_VERSION* = 1;

**static** String *DB\_NAME* = "RSSNews";

**public** DatabaseHelper(Context context) {

**super**(context, *DB\_NAME*, **null**, *DB\_VERSION*);

}

@Override

**public** **void** onCreate(SQLiteDatabase db) {

// **TODO** Auto-generated method stub

String sql = NewsAccess.*getCreateTableSQL*();

db.execSQL(sql);

}

}

* Bước 2: Tạo một lớp **NewsAccess.java** truy cập từ **DatabaseHelper.java**

**public** **class** NewsAccess {

**static** **final** String *TAB\_NAME* = "News";

**static** **final** String *COL\_ID* = "Id";

**static** **final** String *COL\_TITLE* = "Title";

**static** **final** String *COL\_IMG* = "Image";

**static** **final** String *COL\_PUBDATE* = "PubDate";

**static** **final** String *COL\_LINK* = "Link";

Context context;

**private** DatabaseHelper dbHelper;

**public** **static** String getCreateTableSQL() {

String sql = "CREATE TABLE " + *TAB\_NAME* + "(" + *COL\_ID*

+ " integer primary key autoincrement, " + *COL\_TITLE*

+ " text, " + *COL\_PUBDATE* + " text, " + *COL\_LINK* + " text, "

+ *COL\_IMG* + " text)";

**return** sql;

}

**public** NewsAccess(Context context) {

dbHelper = **new** DatabaseHelper(context);

**this**.context = context;

}

}

* Bước 3: Viết hàm Lưu dữ liệu vào database, ta có đoạn code sau:

**private** ContentValues MakeNewsContentValues(News news) {

ContentValues values = **new** ContentValues();

values.put(*COL\_TITLE*, news.getStrTitle());

values.put(*COL\_PUBDATE*, news.getStrDate());

values.put(*COL\_LINK*, news.getStrLink());

values.put(*COL\_IMG*, news.getStrImage());

**return** values;

}

**long** insertNews(News news) {

ContentValues values = MakeNewsContentValues(news);

SQLiteDatabase db = dbHelper.getWritableDatabase();

**long** ok = db.insert(*TAB\_NAME*, **null**, values);

**return** ok;

}

**public** **void** writeNews(News news) {

SQLiteDatabase db = dbHelper.getWritableDatabase();

**boolean** ok = **true**;

**long** position = insertNews(news);

**if** (position == -1)

ok = **false**;

**if** (ok) {

Toast.*makeText*(context, "Bài báo " + news.getStrTitle() +

" đã được lưu vào Database",

Toast.*LENGTH\_LONG*).show();

} **else** {

Toast.*makeText*(

context,

"Lỗi!. Bài báo " + news.getStrTitle() + " này đã có trong Database.",

Toast.*LENGTH\_LONG*).show();

}

db.close();

}

* Bước 4: Tạo một lớp Adapter cho lịch sử có tên HistoryAdapter.java

**public** **class** HistoryAdapter **extends** ArrayAdapter<News> {

Activity context = **null**;

**int** itemlayout;

ArrayList<News> arrNews = **null**;

**public** HistoryAdapter(Activity context, ArrayList<News> objects) {

**super**(context, -1, objects);

// **TODO** Auto-generated constructor stub

**this**.context = context;

//itemlayout = resource;

arrNews = objects;

}

@Override

**public** View getView(**int** position, View convertView, ViewGroup parent) {

// **TODO** Auto-generated method stub

**if**(convertView == **null**){

LayoutInflater inflater = context.getLayoutInflater();

convertView = inflater.inflate(R.layout.*listview\_item\_layout*, **null**);

}

ImageView imgHis = (ImageView) convertView

.findViewById(R.id.*imgItems*);

TextView txtTitleHis = (TextView) convertView.findViewById(R.id.*txtTitle*);

TextView txtDateHis = (TextView) convertView.findViewById(R.id.*txtDate*);

News news = getItem(position);

**if** (news != **null**) {

Picasso.*with*(getContext()).load(news.getStrImage()).into(imgHis);

txtTitleHis.setText(news.getStrTitle());

txtDateHis.setText(news.getStrDate());

}

**return** convertView;

}

}

* Bước 5: Chuyển sang màn hình chi tiết, xử lý sự kiện checkbox để lưu bài viết.

chkSave.setOnCheckedChangeListener(**new** OnCheckedChangeListener() {

**public** **void** onCheckedChanged(CompoundButton arg0, **boolean** arg1) {

// **TODO** Auto-generated method stub

News news;

**if** (chkSave.isChecked()) {

news = **new** News(title, image, date, link);

dba.writeNews(news);

}

}

* Bước 6: Để đọc được một tin tức đã lưu trong database, ta có đoạn code sau:

**public** **void** readContentNews(ArrayList<News> arrNews) {

SQLiteDatabase db = dbHelper.getReadableDatabase();

String query = "Select \* from " + *TAB\_NAME*;

Cursor cur = db.rawQuery(query, **null**);

**if** (cur.moveToFirst()) {

**do** {

News news = **new** News();

news.setId(cur.getInt(cur.getColumnIndex(*COL\_ID*)));

news.setStrTitle(cur.getString(cur.getColumnIndex(*COL\_TITLE*)));

news.setStrDate(cur.getString(cur.getColumnIndex(*COL\_PUBDATE*)));

news.setStrImage(cur.getString(cur.getColumnIndex(*COL\_IMG*)));

news.setStrLink(cur.getString(cur.getColumnIndex(*COL\_LINK*)));

arrNews.add(news);

} **while** (cur.moveToNext());

}

db.close();

}

* Bước 7: Chuyển sang Fragment của lịch sử, ta khai báo đoạn code sau để đọc được dữ liệu từ database ra ngoài listview.

**public** **class** HistoryContent **extends** Fragment {

**private** ListView lvHis;

**private** HistoryAdapter adapter;

**private** ArrayList<News> arrNews = **new** ArrayList<News>();

NewsAccess dba;

@Override

**public** View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// **TODO** Auto-generated method stub

View view = inflater.inflate(R.layout.*fragment\_history\_item\_layout*,

container, **false**);

dba = **new** NewsAccess(getActivity());

lvHis = (ListView) view.findViewById(R.id.*lvHistory*);

adapter = **new** HistoryAdapter(getActivity(), arrNews);

lvHis.setAdapter(adapter);

lvHis.setOnItemClickListener(**new** OnItemClickListener() {

@Override

**public** **void** onItemClick(AdapterView<?> arg0, View arg1, **int** arg2, **long** arg3) {

// **TODO** Auto-generated method stub

Bundle bundle = **new** Bundle();

bundle.putString("title", arrNews.get(arg2).getStrTitle());

bundle.putString("image", arrNews.get(arg2).getStrImage());

bundle.putString("date", arrNews.get(arg2).getStrDate());

bundle.putString("link", arrNews.get(arg2).getStrLink());

Intent intent = **new** Intent(getActivity(), DetailActivity.**class**);

intent.putExtra("news", bundle);

startActivity(intent);

}

});

dba.readContentNews(arrNews);

adapter.notifyDataSetChanged();

**return** view;

}

}

## 3.2 Kiểm thử

* Chương trình gồm có [30 test case](TestReport-Nhom2.xlsx).
* Trong đó:
* Test case PASSED: 27.
* Test case FAILED: 3.

# CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC

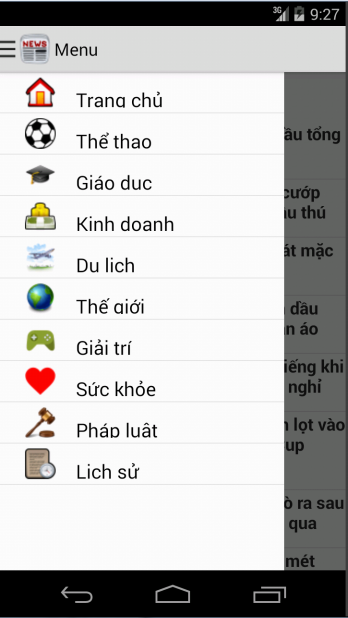
## 4.1 Kết quả đạt được

* Đầu tiên, ta tiến hành chạy ứng dụng thì màn hình chính được hiển thị đầu tiên với các tin tức tổng hợp



Hình 1: Màn hình chính.

* Tiếp theo nhấp vào biểu tượng menu trên thanh ActionBar thì danh sách các chuyên mục được hiển thị ở bên trái màn hình chính.



Hình 2: Danh sách các chuyên mục.

* Sau đó, ta chọn 1 chuyên mục THỂ THAO thì hiển thị danh sách các tin tức liên quan đến THỂ THAO trong màn hình chính.



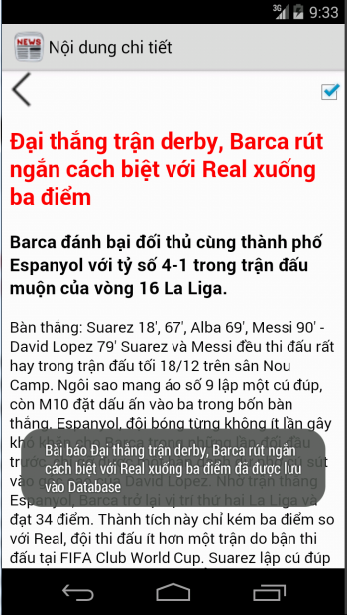
Hình 3: Danh sách tin tức của mục Thể thao

* Giả sử, chọn một tin bất kỳ thì màn hình chính sẽ chuyển sang màn hình chi tiết với nội dung đầy đủ của tin đó.



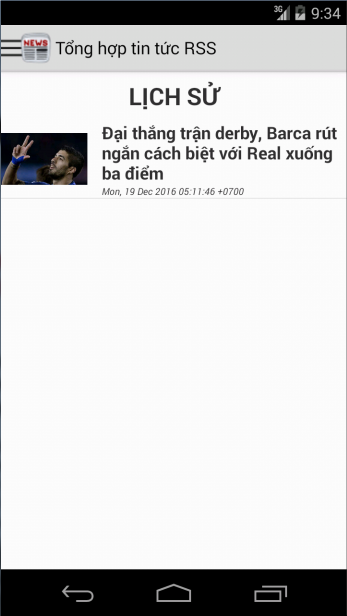
Hình 4: Màn hình nội dung chi tiết.

* Sau đó, ta xử lý chức năng checkbox ghi nhớ tin tức. Khi click vào nút checkbox thì sẽ hiển thị thông báo “Bài báo này đã được lưu vào database” và tin tức này được lưu vào Database



Hình 6: Xử lý nút checkbox ghi nhớ.

* Cuối cùng, ta tiến hành kiểm tra trong danh mục lịch sử đã có tin tức vừa lưu không.



Hình 7: Danh mục lịch sử.

## 4.2 Các kết luận và kiến nghị

* Những điểm đã làm được:
* Sử dụng Slide menu để chia làm nhiều chủ đề, sử dụng tất cả RSS từ trang vnexpress.net.
* Làm chức năng load more cho các tin.
* Lưu bài biết vào database (Lưu offline).
* Đọc được mã nguồn HTML từ thư viện Jsoup.
* Load được ảnh từ Internet bằng thư viện Picasso.
* Những điểm chưa làm được:
* Chưa lấy được ảnh vào màn hình chứa nội dung chi tiết của một tin nào đó.
* Không đọc được File XML
* Không đọc được nội dung từ URL
* Tin tức được ghi nhớ trong chuyên mục lịch sử bị trùng.
* Xử lý sự kiện nút checkbox ghi nhớ ở màn hình chi tiết. Khi ta chọn vào bài viết đó lưu vào database nhưng khi hủy chọn thì nó vẫn lưu vào database (Nghĩa là ta hủy chọn thì bài viết đó sẽ bị hủy bỏ không lưu được).
* Các chức năng bổ sung nếu có thêm thời gian:
* Dưới mỗi bài viết, có thêm các bài liên quan.
* Xây dựng ứng dụng thêm nhiều trang báo hơn.
* Thiết kế đồ họa cho ứng dụng thêm bắt mắt, hài hòa với người sử dụng.

# TÀI LIỆU THAM KHẢO

[1]. Tìm hiểu và nghiên cứu yêu cầu của đề tài:

<http://www.thietkewebsite360.com/tin-tuc/chi-tiet/cach-doc-tin-b%E1%BA%B1ng-rss-cach-su-dung-rss-159/>

<https://vietadsgroup.vn/hoi-dap-la-gi-/rss-la-gi-tim-hieu-ve-rss-la-gi--c62d9904.aspx>

[2]. Cách đọc file XML và hàm đọc nội dung từ URL:

<http://khoapham.vn/KhoaPhamTraining/android/snipet/>

[3]. Lấy các kênh tin tức RSS được vnexpress cung cấp miễn phí:

<http://vnexpress.net/rss>

[4]. Tham khảo cách sử dụng thư viên Jsoup:

<http://tuhocandroid.com/xay-dung-ung-dung-doc-bao-android/>

[5]. Download thư viện hỗ trợ Picasso và Jsoup

<http://square.github.io/picasso/>

<https://jsoup.org/download>