**BÁO CÁO TUẦN 10**

**Họ và tên:** Nguyễn Kỳ Anh  
**MSSV:** 20225793  
 **GR1**  
**Giáo viên hướng dẫn:** Nguyễn Đức Toàn

# UI Thu chi

## Mô tả chức năng

- Chức năng thống kê thu chi cá nhân cho phép người dùng theo dõi tình hình tài chính của mình một cách trực quan và dễ hiểu. Giao diện thể hiện rõ tổng thu nhập, tổng chi tiêu, biểu đồ hình tròn thể hiện tỷ lệ và danh sách các giao dịch cụ thể. Đây là một chức năng quan trọng giúp người dùng quản lý chi tiêu hiệu quả hơn.

## Dữ liệu và tính toán

Dữ liệu thu chi được lưu trong biến Data với cấu trúc gồm tổng thu (TongThu), tổng chi (TongChi), và danhSach các giao dịch.

Map<String, dynamic> Data = {

'TongThu': 0.0,

'TongChi': 0.0,

'danhSach': [

{'Title': 'Chi tiêu', 'Tien': '-500000.0', 'Ngay': '25/05/2025'},

{'Title': 'Thu nhập', 'Tien': '+2000000.0', 'Ngay': '25/05/2025'},

]

};

Mỗi lần cập nhật dữ liệu, ứng dụng gọi hàm \_sumThuChi() để tính lại tổng thu và chi:

void \_sumThuChi() {

double tongThu = 0.0;

double tongChi = 0.0;

for (var danhSach in Data['danhSach']) {

double amount = double.parse(danhSach['Tien'].replaceAll('+', ''));

if (danhSach['Title'] == 'Thu nhập') {

tongThu += amount;

} else {

tongChi += amount.abs();

}

}

setState(() {

Data['TongThu'] = tongThu;

Data['TongChi'] = tongChi;

});

}

## Hiển thị biểu đồ và danh sách giao dịch

Biểu đồ hình tròn sử dụng thư viện fl\_chart để thể hiện tỷ lệ thu và chi:

PieChart(

PieChartData(

sections: [

PieChartSectionData(

color: Colors.yellow,

value: Data['TongThu'],

title: "Thu",

titleStyle: TextStyle(color: Colors.white),

),

PieChartSectionData(

color: Colors.red,

value: Data['TongChi'],

title: "Chi",

titleStyle: TextStyle(color: Colors.white),

),

],

centerSpaceRadius: 40,

sectionsSpace: 2.0,

),

);

Danh sách các giao dịch được hiển thị bằng ListView.builder, thể hiện rõ loại giao dịch, ngày thực hiện và số tiền.

## Giao diện thêm giao dịch mới

Người dùng có thể thêm giao dịch bằng cách nhập số tiền và chọn loại thu hoặc chi. Dữ liệu được trả lại về màn hình chính và cập nhật ngay lập tức:

final amount = \_chonType == 'Thu nhập'

? '+${\_moneyController.text}'

: '-${\_moneyController.text}';

var map = <String, String>{

'Title': \_chonType ?? '',

'Tien': amount,

'Ngay': '${DateTime.now().day}/${DateTime.now().month}/${DateTime.now().year}',

};

Navigator.pop(context, map);

# UI Kiểm kho

## Mô tả chức năng

* Quản lý danh sách sản phẩm theo tên, mã và số lượng.
* Thêm sản phẩm mới thông qua hộp thoại nhập liệu.
* Tìm kiếm nhanh sản phẩm theo từ khóa.
* Xóa sản phẩm không cần thiết khỏi danh sách.

## Lớp dữ liệu BienThanhPhan

class BienThanhPhan {

final String code; // Mã sản phẩm

final String name; // Tên sản phẩm

final int soLuong; // Số lượng sản phẩm

BienThanhPhan({required this.code, required this.name, required this.soLuong});

}

## Khởi tạo và cập nhật dữ liệu.

Dữ liệu được lưu trong 2 danh sách:

* \_danhSachSP: danh sách đầy đủ tất cả sản phẩm.
* \_danhSachSP1: danh sách được lọc theo từ khóa tìm kiếm.

Trong initState(), danh sách lọc ban đầu được gán bằng danh sách gốc, đồng thời gắn sự kiện lắng nghe thay đổi từ TextEditingController tìm kiếm.

@override

void initState() {

super.initState();

\_danhSachSP1 = \_danhSachSP;

\_timKiemController.addListener(danhSachSP); // Tự động lọc khi người dùng nhập

}

## Tìm kiếm sản phẩm.

* Mỗi khi người dùng nhập từ khóa tìm kiếm, hàm danhSachSP() sẽ lọc danh sách dựa trên tên hoặc mã sản phẩm:

void danhSachSP() {

final input = \_timKiemController.text.toLowerCase();

setState(() {

\_danhSachSP1 = \_danhSachSP.where((SP) {

return SP.name.toLowerCase().contains(input) ||

SP.code.toLowerCase().contains(input);

}).toList();

});

}

* Việc sử dụng toLowerCase() giúp loại bỏ phân biệt chữ hoa – chữ thường trong quá trình tìm kiếm.

## Thêm sản phẩm mới

* Sản phẩm được thêm thông qua một hộp thoại AlertDialog. Nếu người dùng nhập đầy đủ tên, mã và số lượng, hàm \_addSP() sẽ được gọi để tạo đối tượng mới:

void \_addSP() {

if (\_nameController.text.isNotEmpty &&

\_codeController.text.isNotEmpty &&

\_countController.text.isNotEmpty) {

setState(() {

\_danhSachSP.add(BienThanhPhan(

code: \_codeController.text,

name: \_nameController.text,

soLuong: int.parse(\_countController.text),

));

\_danhSachSP1 = \_danhSachSP;

});

\_nameController.clear();

\_codeController.clear();

\_countController.clear();

}

}

* Dialog thêm sản phẩm hiển thị 3 trường nhập liệu (tên, mã, số lượng) và 2 nút hành động (Hủy, Thêm). Khi người dùng nhấn "Thêm", hệ thống gọi \_addSP() và đóng hộp thoại.

## Xóa sản phẩm

* Mỗi sản phẩm trong danh sách có biểu tượng thùng rác ở bên phải. Khi người dùng nhấn, hàm \_deleteSP(index) được gọi để xóa phần tử tại vị trí tương ứng:

void \_deleteSP(int index) {

setState(() {

\_danhSachSP.removeAt(index);

\_danhSachSP1 = \_danhSachSP;

});

}

## Hiển hị danh sách sản phẩm.

* Giao diện danh sách được xây dựng bằng ListView.builder, đảm bảo hiệu năng tốt với số lượng lớn dữ liệu:

Expanded(

child: ListView.builder(

itemCount: \_danhSachSP1.length,

itemBuilder: (context, index) {

final SP = \_danhSachSP1[index];

return ListTile(

title: Text(SP.name),

subtitle: Text("Mã: ${SP.code} | SL: ${SP.soLuong}"),

trailing: IconButton(

icon: Icon(Icons.delete, color: Colors.red),

onPressed: () => \_deleteSP(index),

),

);

},

),

)

# UI nhập hàng

## Mô tả chức năng

* Người dùng có thể:
* Thêm hàng mới với thông tin chi tiết như: tên, số lượng, đơn giá...
* Xem danh sách hàng nhập ở dạng bảng.
* Xem thống kê nhanh số lượng hóa đơn hôm nay.
* Lọc theo nhà cung cấp và ngày nhập.

## Dữ liệu danh sách hàng nhập

* Biến \_inventoryItems là một danh sách (List) kiểu Map<String, dynamic> để lưu thông tin các mặt hàng đã nhập.

final List<Map<String, dynamic>> \_inventoryItems = [

{

'id': 1,

'name': 'Gạo ST25',

'quantity': 50,

'price': 20000,

'total': 1000000,

'supplier': 'NCC A',

'date': '28-05-2025',

},

...

];

* Mỗi phần tử trong danh sách là một mặt hàng có các trường:
* id: số thứ tự (auto tăng).
* name: tên hàng.
* quantity: số lượng nhập.
* price: đơn giá.
* total: tổng tiền = số lượng × đơn giá.
* supplier: nhà cung cấp (mặc định NCC A).
* date: ngày nhập (dạng chuỗi).

### Hàm \_showAddItemDialog

* Hàm này dùng để mở form nhập hàng mới. Đây là một AlertDialog cho phép người dùng nhập 3 thông tin chính:
* Tên hàng (name)
* Số lượng (quantity)
* Đơn giá (price)

void \_showAddItemDialog() {

String name = '';

int quantity = 0;

int price = 0;

* Xử lý nhập liệu từ người dùng:
* Sử dụng TextField để nhập liệu.
* Khi thay đổi giá trị, cập nhật vào biến bằng onChanged.

onChanged: (value) => name = value;

onChanged: (value) => quantity = int.tryParse(value) ?? 0;

onChanged: (value) => price = int.tryParse(value) ?? 0;

* Xử lý khi nhấn “Lưu”:
* Kiểm tra hợp lệ: tên không rỗng, số lượng và đơn giá > 0.
* Nếu hợp lệ thì:
  + Thêm dữ liệu vào \_inventoryItems.
  + Tự động gán id mới theo số lượng item hiện có.
  + Tính total = quantity \* price.
  + Lấy ngày hiện tại bằng:

DateTime.now().toString().split(' ')[0];

* Sau khi thêm, đóng hộp thoại và hiện SnackBar thông báo thành công.

ScaffoldMessenger.of(context).showSnackBar(

SnackBar(content: Text('Đã thêm $name thành công!')),

);

## Hiển thị bảng dữ liệu (DataTable)

Dữ liệu trong \_inventoryItems sẽ được hiển thị bằng DataTable. Mỗi dòng (DataRow) sẽ là một mặt hàng, gồm các ô (DataCell) hiển thị:

* STT
* Tên hàng
* Số lượng
* Đơn giá
* Tổng tiền
* Nhà cung cấp
* Ngày nhập

rows: \_inventoryItems.map((item) {

return DataRow(cells: [

DataCell(Text(item['id'].toString())),

DataCell(Text(item['name'])),

DataCell(Text(item['quantity'].toString())),

...

]);

}).toList(),

* Sử dụng SingleChildScrollView để hỗ trợ cuộn ngang nếu bảng bị dài.

## Bộ lọc nhà cung cấp và lọc theo ngày.

 Dùng DropdownButton để chọn nhà cung cấp.

 Nút “Lọc theo ngày” chưa có chức năng, chỉ hiển thị.

DropdownButton<String>(

hint: Text('Tất cả nhà cung cấp'),

items: <String>['NCC A', 'NCC B', 'Tất cả'].map((String value) {

return DropdownMenuItem<String>(

value: value,

child: Text(value),

);

}).toList(),

onChanged: (\_) {},

),

# UI Công cụ.

## Mô tả chức năng

* Màn hình chính giúp người dùng truy cập nhanh đến các module như: thực đơn, hóa đơn, kho hàng, sổ thu chi, khách hàng, nhân viên, báo cáo, v.v.

## Body

* Giao diện chính gồm danh sách các thẻ (Card) được hiển thị bằng ListView.
* Các thẻ đại diện cho từng chức năng:
* Thực đơn
* Phòng bán
* Hóa đơn
* Kho hàng (có thêm 3 mục phụ: Kho hàng, Xuất hủy, Nhập hàng)
* Sổ thu chi, Khách hàng, Nhân viên, Báo cáo
* Mỗi thẻ được tạo bởi hàm \_buildCard(...) để tái sử dụng giao diện linh hoạt.

## Kho hàng

Card(

child: Column(

children: [

ListTile(title: Text('Kho hàng')),

ListTile(title: Text('Xuất hủy')),

ListTile(title: Text('Nhập hàng')),

]

)

* Giao diện chia thành 3 lựa chọn nhỏ bên trong một thẻ Card lớn.
* Tạo hiệu ứng gọn gàng, dễ hiểu cho người dùng.

## Hàm \_buldCard().

Widget \_buildCard({required IconData icon, ...})

* Giúp xây dựng các thẻ chức năng chung, tái sử dụng code.
* Mỗi thẻ có: icon, tiêu đề, mô tả phụ (subtitle), nút điều hướng (chevron\_right).