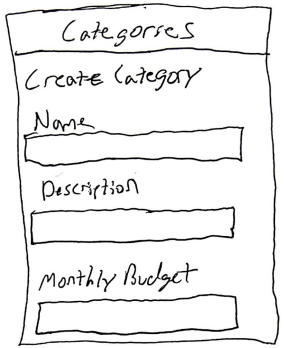
SaveMe

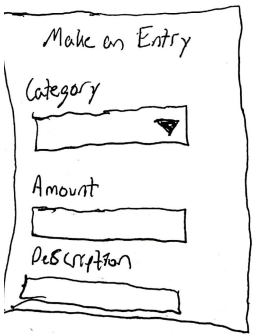
Mobile Application by Ryan Johnson and Aseel Almanahy

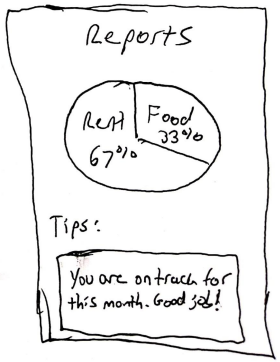
Proposal

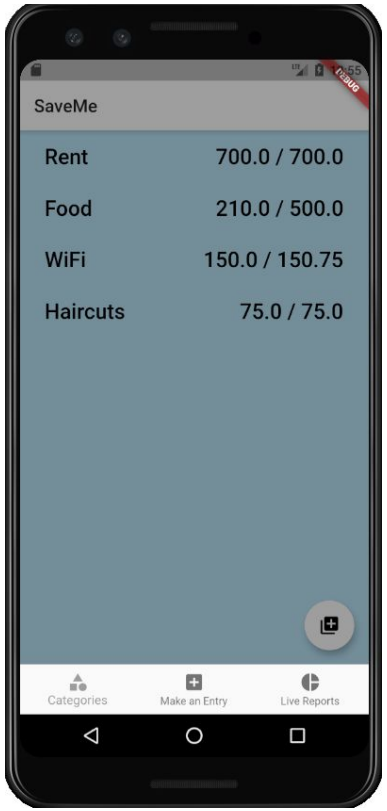
- Our idea was to create a Financial Wellness app, “SaveMe”
- The app allows the user to budget and track their expenses for each month
- Early on, we decided to use Flutter to develop our application
- Originally wanted to use Firebase for the database, but used SQLite

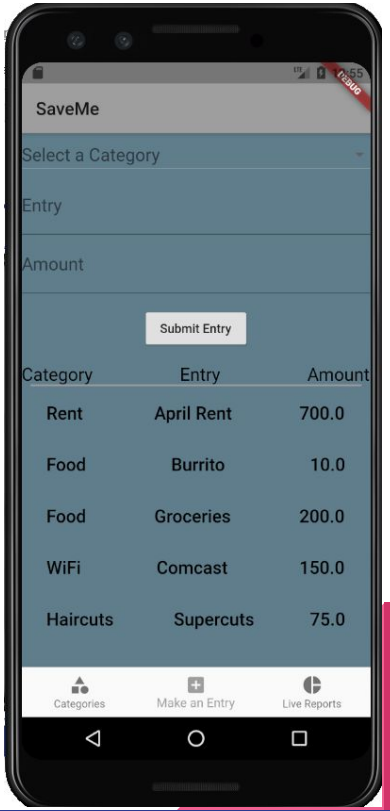
Preliminary Design vs. Actual

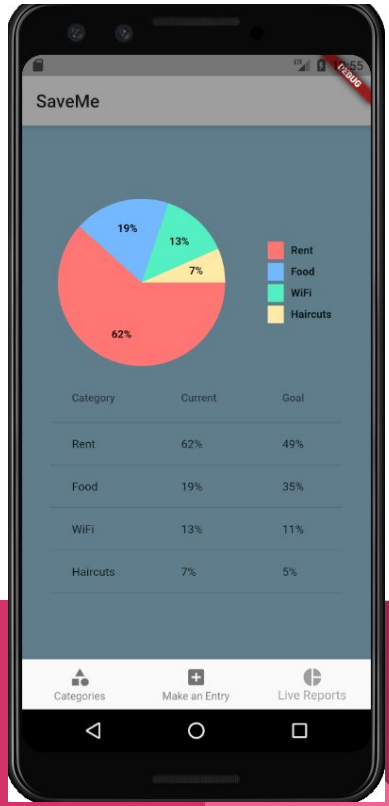






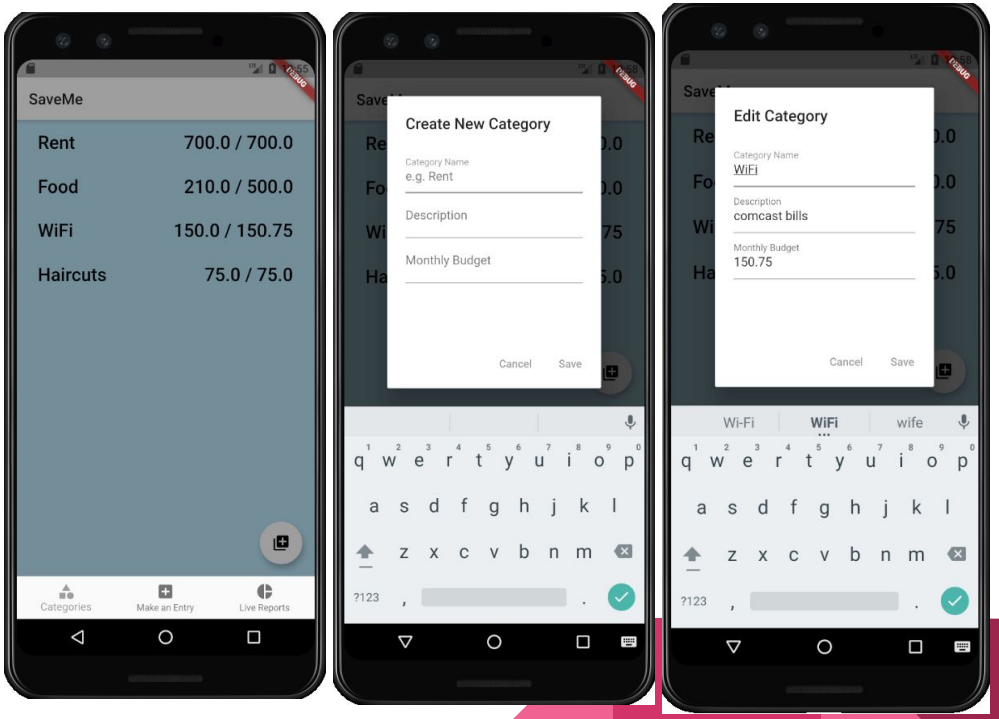






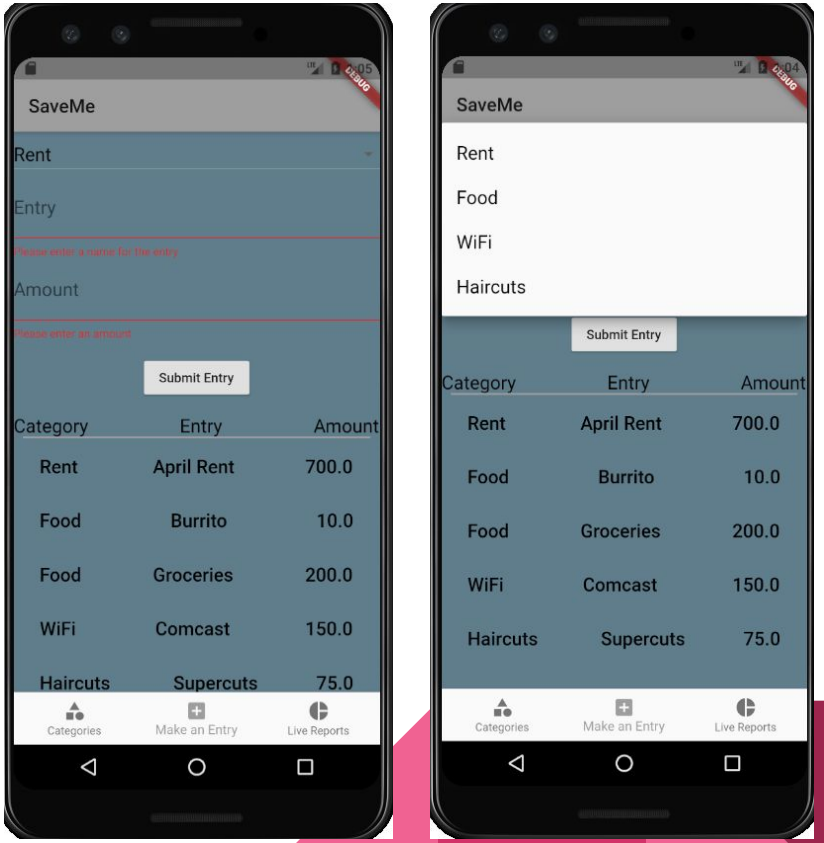
Categories Page Design

- Contains ListView of all categories the user makes
- List is scrollable, and can be deleted by swiping right or left
- To add a category, click the “+” and add the information
- To edit a category, click on the category and edit information



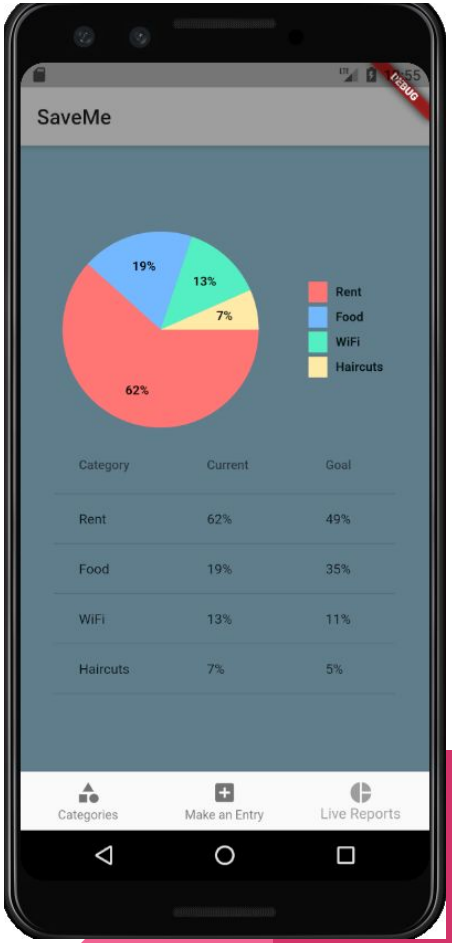
Entries Page Design

- Contains Form Widget for entering expenses
- User selects a category from a drop down, and then enters a name for the entry and an amount
- Form contains validation for missing fields
- List appears below of all entries, and is scrollable



Reports Page Design

- Pie chart displays percentages of amount spent per category vs. total amount spent
- Table below chart (DataTable Widget) shows the current percentage of what you've spent, vs. your goal percentage for that particular category



Implementation

- Flutter utilizes “Widgets” which make up the frontend of the application
- Uses “Dart” programming language
- Widgets can have a wide arrange of layouts such as lists or stacked on one another
- We have a “homeWidget” which includes the Bottom Navigation Bar and the app title/colors
- The other pages are “Widgets” on top of this home widget (i.e. Category, Entry and Reports widgets)

```dart
class _HomeWidgetState extends State<HomeWidget> {

  int _currentIndex = 0;
  final List<Widget> _children = [
    CategoriesPage(),
    EntriesPage(),
    ReportsPage()
  ];

  void onTabTapped(int index) {
    setState(() {
      _currentIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {

    return Scaffold(
      backgroundColor: Colors.blueGrey,
      appBar: AppBar( title: Text(widget.title) ),
      body: _children[_currentIndex],
      bottomNavigationBar: BottomNavigationBar(
        onTap: onTabTapped,
        currentIndex: _currentIndex,
        items: [
          BottomNavigationBarItem(
            icon: new Icon(Icons.category),
            title: new Text('Categories'),
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: new Icon(Icons.add_box),
            title: new Text('Make an Entry'),
          ), // BottomNavigationBarItem
          BottomNavigationBarItem(
            icon: Icon(Icons.pie_chart),
            title: Text('Live Reports')
          ) // BottomNavigationBarItem
        ],
      ), // BottomNavigationBar
    ); // Scaffold
  }
}
```

Implementation cont.

- Our Categories, Entries, and Reports pages all interact with our database
- Database has two tables: Categories and Entries, and is implemented with SQFLite (SQLite for Flutter)
- Categories and Entries pages retrieve from and update the database
- Reports page reads the database values to construct the pie chart and percentages

```
static Future<void> init() async {
  if (_db != null) {
    return;
  }

  try {
    String _path = await getDatabasesPath() + 'example';
    _db = await openDatabase(_path, version: _version, onCreate: onCreate);
  }
  catch (ex) {
    print(ex);
  }
}

static void onCreate(Database db, int version) async {
  await db.execute('CREATE TABLE categories (id INTEGER PRIMARY KEY NOT NULL, name STRING, c
  await db.execute('CREATE TABLE entries (id INTEGER PRIMARY KEY NOT NULL, category STRING,
}

static Future<List<Map<String, dynamic>>> query(String table) async => _db.query(table);

static Future<int> insert(String table, Model model) async =>
    await _db.insert(table, model.toMap());

static Future<int> update(String table, Model model) async =>
    await _db.update(table, model.toMap(), where: 'id = ?', whereArgs: [model.id]);

static Future<int> delete(String table, Model model) async =>
    await _db.delete(table, where: 'id = ?', whereArgs: [model.id]);

static Future getTotalEntries(String categoryName) async {
  List<Map> list = await _db.rawQuery('SELECT SUM(amount) FROM entries');
  print(list[0].values);
  returnResults(list[0].values.toString());
}
```

Helpful Flutter Resources

Pie Chart

https://pub.dev/packages/pie_chart

Form Validation

https://flutter.dev/docs/cookbook/forms/validation

Bottom Navigation Bar

https://api.flutter.dev/flutter/material/BottomNavigationBar-class.html

SQFLite (SQLite for Flutter)

https://flutter.dev/docs/cookbook/persistence/sqlite