



Spice Rack

Michael Carter & Astrid Strohmaier

Contents

- About Spice Rack
- Development Process
- Special Features
- Biggest Bug
- Test Cases
- Application Demo
- Conclusion

About Spice Rack

- App for spice & herb shopping
- Spice Inventory
- Shopping List
- User Profile



User Stories

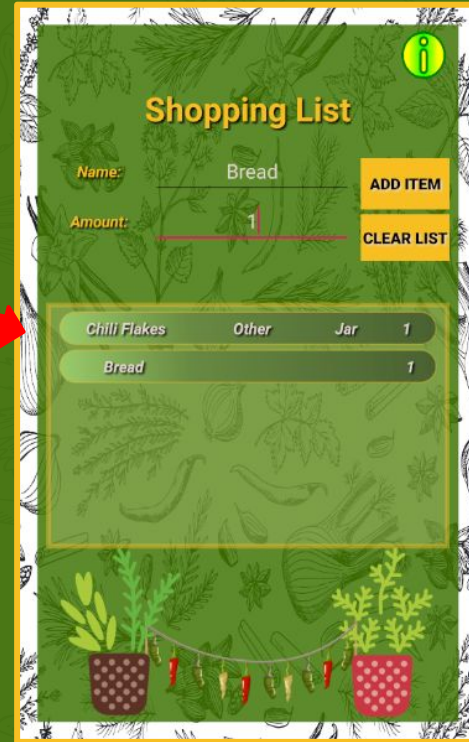


User Stories



The image shows a mobile app interface for a 'Spice Inventory List'. It features a green background with a subtle pattern of various herbs and spices. At the top right is a green circular icon with a white lowercase 'i'. Below the title is a table with four columns: 'Spice name', 'Brand', 'Container', and 'Stock'. The table contains four rows of data. The second row, 'Chilli Flakes', is highlighted with a red oval. A red arrow points from this row to the 'Shopping List' app interface on the right.

Spice name	Brand	Container	Stock
Cayenne Pepper	KMarket	Refill	1
Chilli Flakes	Other	Jar	0
Garlic Salt	Other	Jar	1
Garlic Salt	Santa Maria	Jar	1



The image shows a mobile app interface for a 'Shopping List'. It features a green background with a subtle pattern of various herbs and spices. At the top right is a green circular icon with a white lowercase 'i'. Below the title is a form with two input fields: 'Name:' and 'Amount:'. The 'Name:' field contains the text 'Bread'. To the right of the 'Name:' field is a yellow button labeled 'ADD ITEM'. Below the 'Amount:' field is a yellow button labeled 'CLEAR LIST'. Below the form is a list of items. The first item is 'Chilli Flakes' with a quantity of '1'. The second item is 'Bread' with a quantity of '1'. At the bottom of the screen are two potted plants, one green and one red, with a string of red chili peppers hanging between them.

Name: Bread

Amount: 1

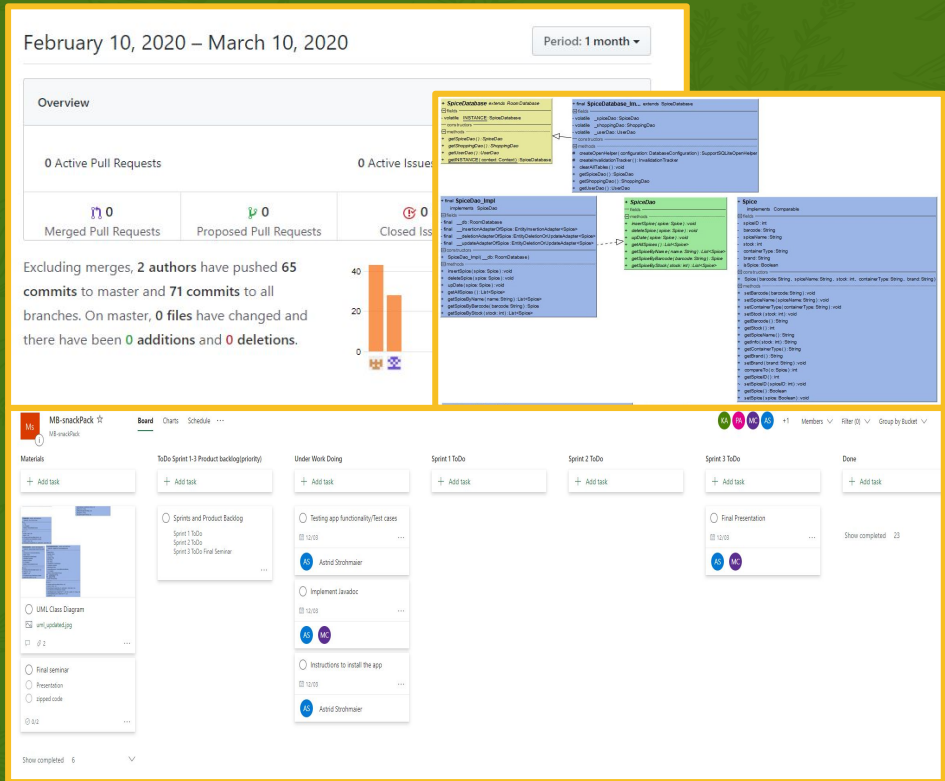
ADD ITEM

CLEAR LIST

Chilli Flakes	Other	Jar	1
Bread			1

Development Process

- Main project management techniques use:
 - SCRUM
 - Microsoft Planner
 - GitHub
 - JavaDocs and UML



Special Features

- Barcode Scanning
- Room Database
- Interactive Spice Inventory
- Interactive Shopping List
- Swipe Gesture Navigation



Biggest Bug - Multi-viewHolder Adapter

- Shopping List RecyclerView.
- Layout defined by View Type.
- Multiple different View Holders
- onItemTouch/Click Listeners.
- Ensuring the correct View Holder is bound too.

```
@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view;
    switch (viewType) {
        case SPICE_NORMAL:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.adapter_mv_spice, parent, attachToRoot: false);
            return new ViewHolderSpice(view, onItemClickListener);
        case SPICE_STRIKE:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.adapter_mv_spice_strikethrough, parent, attachToRoot: false);
            return new ViewHolderSpice(view, onItemClickListener);
        case SHOPPING_NORMAL:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.adapter_mv_shopping, parent, attachToRoot: false);
            return new ViewHolderShopping(view, onItemClickListener);
        case SHOPPING_STRIKE:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.adapter_mv_shopping_strikethrough, parent, attachToRoot: false);
            return new ViewHolderShopping(view, onItemClickListener);
        default:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.home_activity, parent, attachToRoot: false);
            return new ViewHolderSpice(view, onItemClickListener);
    }
}
```

```
@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
    if (holder instanceof ViewHolderSpice) {
        final ViewHolderSpice viewHolderSpice = (ViewHolderSpice) holder;
        viewHolderSpice.spiceName.setText(shoppingLists.get(position).getItemName());
        viewHolderSpice.containerType.setText(shoppingLists.get(position).getContainerType());
        viewHolderSpice.brand.setText(shoppingLists.get(position).getBrand());
        viewHolderSpice.stock.setText(String.valueOf(shoppingLists.get(position).getAmount()));
    } else {
        final ViewHolderShopping viewHolderShopping = (ViewHolderShopping) holder;
        viewHolderShopping.shoppingName.setText(shoppingLists.get(position).getItemName());
        viewHolderShopping.amount.setText(String.valueOf(shoppingLists.get(position).getAmount()));
    }
}
```

```
static class ViewHolderSpice extends RecyclerView.ViewHolder implements View.OnClickListener {
    TextView spiceName, stock, containerType, brand;
    OnItemClickListener onItemClickListener;

    /**
     * sets the Layout of the object ready for data to be assigned.
     */
    @SuppressWarnings("ViewHolderView")
    ViewHolderSpice(View itemView, OnItemClickListener onItemClickListener) {
        super(itemView);
        this.onItemClickListener = onItemClickListener;
        spiceName = itemView.findViewById(R.id.rvSpiceName);
        brand = itemView.findViewById(R.id.rvBrand);
        containerType = itemView.findViewById(R.id.rvContainerType);
        stock = itemView.findViewById(R.id.rvStock);
        itemView.setOnClickListener(this);
    }

    /**
     * gets the position of the object clicked within the recycler view list.
     */
    @Override
    public void onClick(View v) { onItemClickListener.onItemClick(getAdapterPosition()); }
}

static class ViewHolderShopping extends RecyclerView.ViewHolder implements View.OnClickListener {
    TextView shoppingName, amount;
    OnItemClickListener onItemClickListener;

    /**
     * sets the Layout of the object ready for data to be assigned.
     */
    @SuppressWarnings("ViewHolderView")
    ViewHolderShopping(View itemView, OnItemClickListener onItemClickListener) {
        super(itemView);
        this.onItemClickListener = onItemClickListener;
        shoppingName = itemView.findViewById(R.id.rvShoppingName);
        amount = itemView.findViewById(R.id.rvShoppingAmount);
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) { onItemClickListener.onItemClick(getAdapterPosition()); }
}
```


Testing

- 4 test phases
 - Development process
 - More devices/emulators
 - 2 rounds of [test cases](#)
- Findings:
 - Earlier testing with multiple devices
 - Minor bugs
 - Shorter feedback form
 - Positive feedback

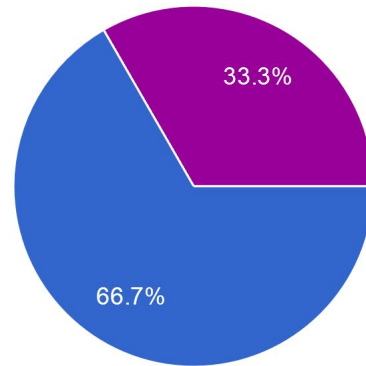


The screenshot shows the 'Spice Inventory Editor' app interface. It features a green background with a pattern of various spices. At the top right is a green circular information icon. The title 'Spice Inventory Editor' is displayed in yellow. Below the title are several input fields: 'Barcode number:' with a placeholder 'Barcode here', 'Name:' with a placeholder 'Name here', 'Current stock:' with a placeholder 'Stock here', 'Container type:' with a dropdown menu showing 'Select One...', and 'Brand:' with a dropdown menu showing 'Select One...'. Below these fields are three yellow buttons labeled 'DELETE SPICE', 'UPDATE', and 'ADD SPICE'. At the bottom, there is an illustration of a mortar and pestle, a garlic bulb, and two red chili peppers, next to a yellow circular button containing a barcode.

Testing

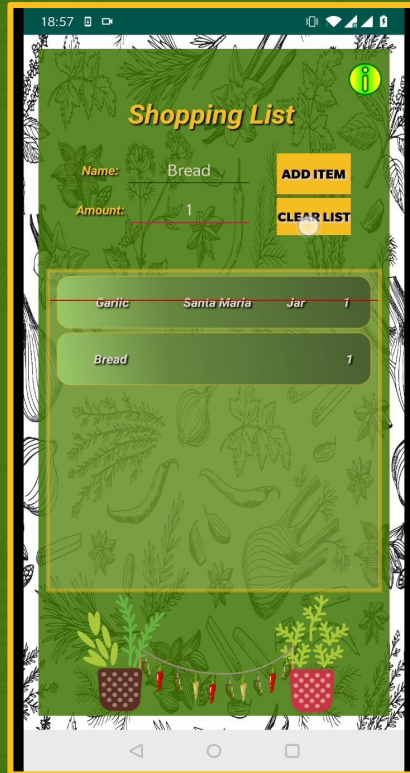
From the picture of fonts provided in the instructions, which would like to see as the main font style?

3 responses



- 1. Default Android - Roboto
- 2. Caladea Regular
- 3. PTS Sans Narrow Regular
- 4. TradeWinds Regular
- 5. Lemonada Regular
- 6. Oregano Regular
- 7. Unkempt Regular

Application Demo



Conclusion

- Simplified herb and spice shopping
- Overcame obstacles
- Learning process
- Rewarding experience
- Future
 - Encrypted passwords
 - Online database
 - Smart spice level measurements

Links and Acknowledgements

- Special thanks to Kinga Koterska for the initial application design.
- [GitHub](#)
 - References for barcode scanner library and images in the application can be found here in the README file.
- [JavaDoc](#)
- [Spice Rack - Test instructions](#)
- [Test Cases - Feedback](#)
- [Test Cases - Feedback old form](#)
- [Barcode Image](#)



Thank you