*TastyTracker Developer’s Guide*

Julie White (2832202)

Table of Contents

[Java Classes 3](#_Toc184154334)

[EditItemActivity 3](#_Toc184154335)

[foodDBAdapter 5](#_Toc184154336)

[foodItem 6](#_Toc184154337)

[HouseholdListAdapter 7](#_Toc184154338)

[InitialActivity 8](#_Toc184154339)

[InventoryActivity 9](#_Toc184154340)

[InventoryAdapter 10](#_Toc184154341)

[LoginActivity 11](#_Toc184154342)

[ManageHouseholdActivity 12](#_Toc184154343)

[ManageRequestsActivity 13](#_Toc184154344)

[RegisterActivity 14](#_Toc184154345)

[RequestsListAdapter 15](#_Toc184154346)

[ShoppingActivity 16](#_Toc184154347)

[shoppingListItem 17](#_Toc184154348)

[SplashActivity 18](#_Toc184154349)

[User 18](#_Toc184154350)

[userInfoDBAdapter 19](#_Toc184154351)

[UserSession 20](#_Toc184154352)

[XML Files 22](#_Toc184154353)

[activity\_edit\_item 22](#_Toc184154354)

[activity\_initial 23](#_Toc184154355)

[activity\_inventory 23](#_Toc184154356)

[activity\_login 23](#_Toc184154357)

[activity\_manage\_household 24](#_Toc184154358)

[activity\_manage\_requests 24](#_Toc184154359)

[activity\_register 25](#_Toc184154360)

[activity\_shopping 25](#_Toc184154361)

[activity\_splash 25](#_Toc184154362)

[household\_list\_layout 26](#_Toc184154363)

[inventory\_list\_layout 26](#_Toc184154364)

[requests\_list\_layout 27](#_Toc184154365)

[shopping\_list\_layout 27](#_Toc184154366)

# Java Classes

Java classes in the TastyTracker project primarily handle logic and functionality. They define the behavior of the application and are responsible for:

1. User Interactions: Handling user inputs and actions, such as logging in, registering, editing items, and managing permissions.
2. Business Logic: Implementing rules and features, like adding items to inventory or approving requests.
3. Data Management: Interacting with SQLite databases to store and retrieve data.
4. Adapters: Formatting and displaying data in lists or grids, such as inventory items, shopping lists, or household members.
5. Activity Navigation: Controlling the flow between different screens (e.g., from login to inventory view).

## EditItemActivity

The EditItemActivity class is responsible for managing the process of adding, editing, or deleting items in the inventory or shopping list. It provides the logic behind user interactions with the activity\_edit\_item.xml layout.

Key Role:

1. Adding a New Item: Enables users to input item details (name, quantity, and unit) and save them to the inventory or shopping list.
2. Editing an Existing Item: Allows users to modify the attributes of an existing item.
3. Deleting an Item: Provides the option to remove an item entirely from the inventory or shopping list.

Important Variables:

1. UI Components

- itemNameEditText: Captures the item's name input.

- itemQuantityEditText: Captures the item's quantity input.

- itemUnitEditText: Captures the item's unit input.

- saveButton: Triggers the save operation.

- cancelButton: Cancels the action and returns to the previous activity.

- deleteButton: Deletes the current item.

1. Intent Extras

- MODE: Determines whether the activity is in 'ADD' or 'EDIT' mode.

- RETURN: Specifies the source activity to navigate back to after the operation.

- TOADD: Indicates where the item should be saved.

- HOUSEHOLD\_ID: Specifies the household's unique ID for database operations.

- ITEM\_NAME, ITEM\_QUANTITY, ITEM\_UNIT: Pass details of the item being edited.

1. Database Adapter

- foodDBAdapter dbAdapter: Interacts with the database for adding, editing, or deleting items.

Key Methods

- onCreate(): Initializes the UI components. Checks the MODE intent extra to set the activity for adding or editing. Populates the fields with item details if in 'EDIT' mode.

- saveItem(): Validates user input (e.g., checks if fields are empty). Determines whether to insert a new item or update an existing one in the database.

- deleteItem(): Deletes the current item from the database. Confirms the action with the user before deletion.

- cancelAction(): Returns to the previous activity without making changes.

## foodDBAdapter

The foodDBAdapter class is responsible for interacting with the SQLite database to manage inventory, shopping lists, and request items for each household. It provides methods for inserting, updating, retrieving, and deleting records.

Key Role:

* + Manage household-specific databases for inventory, shopping lists, and requests.
  + Provide database connection management, including opening and closing connections.
  + Handle CRUD (Create, Read, Update, Delete) operations for all item types.
  + Support data queries for inventory and shopping list displays.

Important Variables

1. Database Configuration
   1. - DATABASE\_NAME: Name of the SQLite database.
   2. - DATABASE\_VERSION: Current version of the database, used for schema upgrades.
   3. - INVENTORY\_TABLE, SHOPPING\_LIST\_TABLE, REQUESTS\_TABLE: Table names for storing data.
2. Database Helper

The inner DatabaseHelper class manages database creation and schema upgrades. It extends SQLiteOpenHelper to provide lifecycle management.

Key Methods

* open(int householdID): Opens a connection to the database for the specified household.
* close(): Closes the current database connection.
* insertItem(int householdID, String name, double quantity, String unit, boolean shopped, String type): Adds a new item to the inventory, shopping list, or requests table based on the type parameter.
* updateItem(int householdID, String name, double quantity, String unit, String type): Updates an existing item's attributes in the specified table.
* deleteItem(int householdID, String name, String type): Removes an item from the specified table.
* getInventoryOrRequestItems(int householdID, String tableSuffix): Retrieves items from the inventory or requests table for a specific household.
* getShoppingListItems(int householdID): Retrieves all items from the shopping list for a household.
* getShoppedItems(int householdID): Returns items marked as 'shopped' in the shopping list.
* resetShoppedItems(int householdID): Resets the 'shopped' status for all shopping list items in a household.

## foodItem

The foodItem class represents individual items in the inventory. It provides attributes and methods for managing item details such as name, quantity, and unit. This class serves as a base model for item-related operations in TastyTracker.

Key Role

* Encapsulate the data for individual items, including name, quantity, and unit.
* Serve as a base class for specialized item types like shoppingListItem.
* Facilitate data exchange between the app's activities and database operations.

Important Variables

* String name: The name of the item (e.g., 'Apples', 'Milk').
* double quantity: The quantity of the item (e.g., 2.5 for '2.5 liters').
* String unit: The unit of measurement for the item (e.g., 'kg', 'liters').

Key Methods

* foodItem(String name, double quantity, String unit): Constructor to initialize a food item with specified attributes.
* String getName(): Retrieves the name of the item.
* double getQuantity(): Retrieves the quantity of the item.
* String getUnit(): Retrieves the unit of measurement for the item.
* void setName(String name): Updates the item's name.
* void setQuantity(double quantity): Updates the item's quantity.
* void setUnit(String unit): Updates the item's unit.

## HouseholdListAdapter

The HouseholdListAdapter class is responsible for displaying and managing household member details in a ListView. It connects the data for each household member, such as username and permissions, with the corresponding UI components in the household\_list\_layout.xml layout.

Key Role:

* Bind household member data (username, permissions) to the UI components.
* Enable the head of household to modify permissions using a switch.
* Ensure that the ListView updates dynamically when data changes.

Important Variables:

* Context mContext: The context from the calling activity.
* ArrayList<User> mList: The list of household members to display.
* userInfoDBAdapter userInfoDB: Database adapter to handle user permission updates.

Key Methods:

* HouseholdListAdapter(Context context, ArrayList<User> list): Constructor to initialize the adapter with a context and list of users.
* getView(int position, View convertView, ViewGroup parent): Responsible for binding data from the User object at the given position to the ListView row.
* onCheckedChanged(CompoundButton buttonView, boolean isChecked): Updates the permissions for a user when the switch is toggled.

## InitialActivity

The InitialActivity class serves as the entry point for the TastyTracker application. It presents users with the option to either log in to an existing account or register a new one. This activity connects with activity\_initial.xml to provide the corresponding user interface.

Key Role:

* Display options for users to log in or register.
* Navigate to the LoginActivity or RegisterActivity based on user selection.
* Provide an engaging and welcoming first impression of the app.

Important Variables:

* Button loginButton: Navigates to the LoginActivity when clicked.
* Button registerButton: Navigates to the RegisterActivity when clicked.

Key Methods

* onCreate(Bundle savedInstanceState): Initializes the UI components and sets click listeners for the buttons.
* navigateToLogin(): Opens the LoginActivity when the login button is clicked.
* navigateToRegister(): Opens the RegisterActivity when the register button is clicked.

## InventoryActivity

The InventoryActivity class is responsible for displaying and managing the household inventory. It interacts with the database to fetch inventory items and allows users to add, edit, or delete items. This activity connects with activity\_inventory.xml to provide the corresponding user interface.

Key Role:

* Display the current inventory items for the household.
* Allow users to navigate to other features, such as adding items or viewing the shopping list.
* Provide options for editing or removing items in the inventory.
* Ensure data is synchronized with the household's database.

Important Variables

* ListView listView: Displays the list of inventory items.
* foodDBAdapter dbAdapter: Handles database operations for inventory management.
* ImageButton addButton: Navigates to the EditItemActivity for adding a new item.
* ImageButton shoppingListButton: Navigates to the ShoppingActivity.

Key Methods:

* onCreate(Bundle savedInstanceState): Initializes the activity, sets up the UI components, and loads inventory data.
* loadInventoryItems(): Fetches inventory items from the database and populates the ListView.
* navigateToAddItem(): Opens the EditItemActivity to allow users to add a new item to the inventory.
* navigateToShoppingList(): Opens the ShoppingActivity to display the household's shopping list.

## InventoryAdapter

The InventoryAdapter class is responsible for managing the display of inventory items in a ListView. It bridges the data for inventory items with the UI components defined in inventory\_list\_layout.xml, ensuring that each item's details are rendered correctly in the app.

Key Role:

* Bind inventory item data (name, quantity, unit) to the UI components of each row in the ListView.
* Provide interactive options for editing or adding items to the shopping list.
* Ensure the ListView dynamically reflects changes in the underlying data.

Important Variables

* Context mContext: The context of the activity that created the adapter.
* ArrayList<foodItem> mList: The list of inventory items to display in the ListView.
* int householdID: The unique ID of the household, used for database interactions.

Key Methods:

* InventoryAdapter(Context context, ArrayList<foodItem> list, int householdID): Constructor to initialize the adapter with the context, item list, and household ID.
* getView(int position, View convertView, ViewGroup parent): Binds the data from a foodItem object at the specified position to the corresponding row in the ListView.
* editItem(String itemName, double quantity, String unit): Navigates to the EditItemActivity to allow users to edit the selected inventory item.
* addToShoppingList(String itemName, double quantity, String unit): Adds the selected inventory item to the shopping list and updates the database.

## LoginActivity

The LoginActivity class is responsible for authenticating users attempting to log in to the TastyTracker application. It validates user credentials against the database and manages the transition to the InventoryActivity upon successful login. This activity connects with activity\_login.xml to provide the corresponding user interface.

Key Role:

* Validate the user's login credentials (username and password).
* Authenticate users by checking their credentials against the database.
* Navigate to the InventoryActivity upon successful login.
* Provide error feedback for invalid login attempts.

Important Variables:

* EditText usernameEditText: Captures the user's entered username.
* EditText passwordEditText: Captures the user's entered password.
* Button loginButton: Triggers the login process when clicked.
* userInfoDBAdapter userInfoDB: Handles database operations for user authentication.

Key Methods:

* onCreate(Bundle savedInstanceState): Initializes the activity and sets up the UI components.
* attemptLogin(): Validates user inputs, checks credentials against the database, and handles navigation upon success or error.
* showLoginError(): Displays an error message when the entered credentials are invalid.
* navigateToInventory(int householdID): Transitions to InventoryActivity, passing the household ID for data retrieval.

## ManageHouseholdActivity

The ManageHouseholdActivity class is responsible for enabling the Head of Household to manage household members and their permissions. It displays a list of household members and provides interactive options to adjust their access rights. This activity connects with activity\_manage\_household.xml to provide the corresponding user interface.

Key Role:

* Display the list of household members, including their usernames and permissions.
* Allow the Head of Household to update member permissions interactively using switches.
* Provide navigation to the ManageRequestsActivity for handling item requests.
* Integrate with the database to persist permission changes.

Important Variables

* ListView listView: Displays the list of household members.
* ImageButton backButton: Navigates back to the previous activity.
* Button requestsButton: Opens the ManageRequestsActivity.
* userInfoDBAdapter userInfoDB: Handles database operations for user management.
* int householdID: Unique identifier for the household, used for database queries.

Key Methods

* onCreate(Bundle savedInstanceState): Initializes the activity, sets up UI components, and populates the household member list.
* loadHouseholdMembers(): Fetches household members from the database and displays them in the ListView.
* updatePermissions(String username, String newPermissions): Updates the specified user's permissions in the database.
* navigateToRequests(): Opens the ManageRequestsActivity to handle item requests from household members.

## ManageRequestsActivity

The ManageRequestsActivity class is responsible for displaying and managing item requests made by household members. It allows the Head of Household to approve or deny these requests via a user-friendly interface. This activity connects with activity\_manage\_requests.xml to provide the corresponding user interface.

Key Role:

* Display a list of pending item requests from household members.
* Enable the Head of Household to approve or deny requests interactively.
* Integrate with the database to update the status of requests.
* Provide navigation back to the ManageHouseholdActivity.

Important Variables

* RecyclerView recyclerView: Displays the list of item requests in a scrollable view.
* ImageButton backButton: Navigates back to the ManageHouseholdActivity.
* RequestsListAdapter adapter: Binds the request data to the RecyclerView.
* foodDBAdapter dbAdapter: Handles database operations for managing requests.
* int householdID: Unique identifier for the household, used for database queries.

Key Methods:

* onCreate(Bundle savedInstanceState): Initializes the activity, sets up UI components, and populates the RecyclerView with requests.
* loadRequests(): Fetches pending requests from the database and updates the RecyclerView.
* approveRequest(int requestId): Marks a request as approved in the database and updates the inventory.
* denyRequest(int requestId): Marks a request as denied in the database.
* navigateBack(): Returns to the ManageHouseholdActivity.

## RegisterActivity

The RegisterActivity class is responsible for managing user registration in the TastyTracker application. It allows new users to create an account and optionally join an existing household by providing a Household ID. This activity connects with activity\_register.xml to provide the corresponding user interface.

Key Role

* Capture and validate user inputs, including username, password, and optional Household ID.
* Create a new user account in the database.
* Associate the user with an existing household if a valid Household ID is provided.
* Provide error feedback for invalid or incomplete registration attempts.
* Navigate to the appropriate activity upon successful registration.

Important Variables

* EditText usernameEditText: Captures the user's entered username.
* EditText passwordEditText: Captures the user's entered password.
* EditText householdIDEditText: Captures the optional Household ID for joining an existing household.
* Button registerButton: Triggers the registration process when clicked.
* userInfoDBAdapter userInfoDB: Handles database operations for user registration.

Key Methods

* onCreate(Bundle savedInstanceState): Initializes the activity, sets up UI components, and assigns event listeners.
* attemptRegistration(): Validates user inputs, creates a new user account, and handles household association if applicable.
* validateInputs(): Ensures that the username and password meet basic requirements.
* showRegistrationError(String message): Displays error messages for failed registration attempts.
* navigateToNextActivity(): Directs the user to the InitialActivity or another appropriate screen after successful registration.

## RequestsListAdapter

The RequestsListAdapter class is responsible for managing the display of item requests in a RecyclerView. It binds data related to item requests, such as name, quantity, and unit, to the corresponding UI components defined in requests\_list\_layout.xml.

Key Role:

* Bind request data (name, quantity, unit) to the UI components in each RecyclerView item.
* Ensure the RecyclerView dynamically reflects updates to the list of requests.
* Provide a user-friendly view for managing item requests.

Important Variables

* Context mContext: The context from the calling activity.
* ArrayList<foodItem> mList: The list of item requests to be displayed.
* ViewHolder: Holds references to the UI components for each RecyclerView item (e.g., TextViews for name, quantity, and unit).

Key Methods

* RequestsListAdapter(Context context, ArrayList<foodItem> list): Constructor to initialize the adapter with the context and data list.
* onCreateViewHolder(ViewGroup parent, int viewType): Inflates the layout for each item in the RecyclerView.
* onBindViewHolder(ViewHolder holder, int position): Binds data from the foodItem object at the given position to the corresponding UI components.
* getItemCount(): Returns the total number of requests in the data list.

## ShoppingActivity

The ShoppingActivity class is responsible for displaying and managing the shopping list for a household. It provides options to add items to the shopping list, mark items as purchased, and navigate back to the inventory. This activity connects with activity\_shopping.xml to provide the corresponding user interface.

Key Role

* Display the household's shopping list in a ListView.
* Allow users to mark items as purchased and update the inventory accordingly.
* Enable navigation to add new items to the shopping list.
* Provide seamless transition back to the InventoryActivity.

Important Variables

* ListView listView: Displays the shopping list items.
* foodDBAdapter dbAdapter: Handles database operations for managing the shopping list.
* ImageButton addItem: Navigates to the EditItemActivity to add a new item to the shopping list.
* ImageButton markAsShopped: Processes items marked as purchased, moving them to the inventory.
* ImageButton backToInventory: Navigates back to the InventoryActivity.
* int householdID: Unique identifier for the household, used for database queries.

Key Methods

* onCreate(Bundle savedInstanceState): Initializes the activity, sets up the UI components, and loads the shopping list data.
* loadShoppingListItems(): Fetches shopping list items from the database and populates the ListView.
* markItemsAsShopped(): Processes checked items, adds them to the inventory, and removes them from the shopping list.
* navigateToAddItem(): Opens the EditItemActivity to allow users to add a new item to the shopping list.
* navigateToInventory(): Returns to the InventoryActivity.

## shoppingListItem

The shoppingListItem class represents individual items in the shopping list. It extends the foodItem class to include additional attributes and methods specifically related to shopping list functionality, such as tracking whether an item has been purchased.

Key Role

* Encapsulate the data for shopping list items, including name, quantity, unit, and purchase status.
* Provide getter and setter methods for managing item attributes.
* Extend the functionality of foodItem to include attributes unique to shopping list items.

Important Variables

* boolean shopped: Indicates whether the item has been marked as purchased.

Key Methods

* shoppingListItem(String name, double quantity, String unit, boolean shopped): Constructor to initialize a shopping list item with the specified attributes.
* boolean getShopped(): Retrieves the purchase status of the item.
* void setShopped(boolean newShopped): Updates the purchase status of the item.

## SplashActivity

The SplashActivity class is responsible for displaying the splash screen of the TastyTracker application. It provides an initial loading screen with the app's logo and transitions to the InitialActivity after a brief delay. This activity connects with activity\_splash.xml to provide the corresponding user interface.

Key Role

* Display the splash screen with the app logo during the app startup.
* Introduce a delay to simulate loading before transitioning to the next activity.
* Navigate to the InitialActivity after the splash screen is displayed.

Important Variables

* int SPLASH\_DISPLAY\_LENGTH: Duration (in milliseconds) for which the splash screen is displayed.
* Handler handler: Manages the delay before transitioning to the next activity.

Key Methods

* onCreate(Bundle savedInstanceState): Initializes the activity, sets up the UI components, and starts the handler for the splash screen duration.
* transitionToInitialActivity(): Navigates to the InitialActivity after the splash duration is completed.

## User

The User class represents a user of the TastyTracker application. It encapsulates user-specific details such as username, password, household ID, and permissions. This class serves as a data model for managing user information throughout the app.

Key Role:

* Encapsulate user details including username, password, household ID, and permissions.
* Provide getter and setter methods for accessing and modifying user attributes.
* Serve as a standard data structure for passing user information across activities.

Important Variables

* String username: The username of the user.
* String password: The user's password
* int householdID: The unique ID of the household the user belongs to.
* String permissions: The permissions assigned to the user (e.g., Head of Household, Member with Privileges).

Key Methods:

* User(String username, String password, int householdID, String permissions): Constructor to initialize a user with specified attributes.
* String getUsername(): Retrieves the username of the user.
* String getPassword(): Retrieves the password of the user.
* int getHouseholdID(): Retrieves the household ID associated with the user.
* String getPermissions(): Retrieves the permissions assigned to the user.
* void setUsername(String username): Updates the username of the user.
* void setPassword(String password): Updates the password of the user.
* void setHouseholdID(int householdID): Updates the household ID of the user.
* void setPermissions(String permissions): Updates the permissions of the user.

## userInfoDBAdapter

The userInfoDBAdapter class handles database operations related to user management. It provides methods for creating user accounts, validating login credentials, and managing user permissions. This class facilitates interaction with the SQLite database for user-related functionality.

Key Role:

* Manage user-related database tables, including creating and updating them.
* Insert new user records during registration.
* Validate user credentials during login.
* Retrieve and update user information, such as permissions and household ID.

Important Variables

* String DATABASE\_NAME: The name of the database for user-related data.
* int DATABASE\_VERSION: The version of the database, used for schema updates.
* String TABLE\_USERS: The name of the table storing user information.
* SQLiteDatabase db: The database instance used for executing queries.

Key Methods:

* open(): Opens a writable database connection for performing operations.
* close(): Closes the database connection.
* insertUser(String username, String password, int householdID, String permissions): Adds a new user to the database.
* validateUser(String username, String password): Validates the user's login credentials.
* getUserPermissions(String username): Retrieves the permissions assigned to the specified user.
* updatePermissions(String username, String newPermissions): Updates the permissions for a user.
* updateHouseholdID(String username, int newHouseholdID): Updates the household ID for a user.

## UserSession

The UserSession class manages the session of the currently logged-in user in the TastyTracker application. It provides methods to store and retrieve user details such as username, household ID, and permissions, ensuring consistent access to user information across activities.

Key Role:

* Store session data for the currently logged-in user.
* Provide methods to access user details like username, household ID, and permissions.
* Facilitate session management, ensuring that user information is readily available to other components.

Important Variables:

* String username: The username of the currently logged-in user.
* int householdID: The household ID associated with the logged-in user.
* String permissions: The permissions assigned to the user (e.g., Head of Household, Member with Privileges).

Key Methods:

* init(String username, int householdID, String permissions): Initializes the session with the logged-in user's details.
* String getUsername(): Retrieves the username of the logged-in user.
* int getHouseholdID(): Retrieves the household ID of the logged-in user.
* String getPermissions(): Retrieves the permissions assigned to the logged-in user.
* void setPermissions(String permissions): Updates the user's permissions during the session.
* boolean isSessionActive(): Checks if a session is currently active.

# XML Files

XML files in the TastyTracker project define the user interface (UI). They are responsible for:

1. Screen Design: Structuring the layout of each screen, including buttons, text fields, and other UI elements.
2. Styling: Setting the visual appearance, such as colors, fonts, padding, and margins.
3. Interaction Elements: Placing interactive components (e.g., buttons, text inputs) that are linked to Java code for functionality.
4. Accessibility: Specifying descriptions, labels, and other attributes for accessibility features.

## activity\_edit\_item

The activity\_edit\_item.xml file defines the user interface for the EditItemActivity. It provides input fields and buttons for adding, editing, or deleting inventory or shopping list items. This layout ensures a structured and intuitive interface for managing item details.

Key UI Components

* EditText itemNameEditText: Allows the user to input the item's name.
* EditText itemQuantityEditText: Allows the user to input the item's quantity.
* EditText itemUnitEditText: Allows the user to input the item's unit of measurement.
* Button saveButton: Saves the entered details to the database.
* Button cancelButton: Cancels the operation and returns to the previous screen.
* Button deleteButton: Deletes the current item from the database.

## activity\_initial

The activity\_initial.xml file defines the user interface for the InitialActivity. It provides buttons for navigating to the login or registration screens, ensuring a smooth user experience at the entry point of the TastyTracker application.

Key UI Components

- Button loginButton: Navigates to the LoginActivity when clicked.

- Button registerButton: Navigates to the RegisterActivity when clicked.

- ImageView splashImage: Displays the app's logo for branding and aesthetic purposes.

## activity\_inventory

The activity\_inventory.xml file defines the user interface for the InventoryActivity. It displays a list of inventory items and provides options to add new items, edit existing ones, or navigate to other features such as the shopping list.

Key UI Components

* ListView listView: Displays the list of inventory items.
* ImageButton addButton: Allows users to navigate to the EditItemActivity for adding new items.
* ImageButton shoppingListButton: Navigates to the ShoppingActivity.

## activity\_login

The activity\_login.xml file defines the user interface for the LoginActivity. It provides input fields for entering a username and password, along with a login button to authenticate users.

Key UI Components

* EditText usernameEditText: Captures the user's entered username.
* EditText passwordEditText: Captures the user's entered password.
* Button loginButton: Triggers the login process when clicked.
* ImageView splashImage: Displays the app's logo for branding and visual appeal.

## activity\_manage\_household

The activity\_manage\_household.xml file defines the user interface for the ManageHouseholdActivity. It displays a list of household members along with their permissions, and provides options to update permissions or navigate to manage item requests.

Key UI Components

* ListView householdListView: Displays the list of household members and their permissions.
* ImageButton backButton: Allows users to navigate back to the previous activity.
* Button requestsButton: Opens the ManageRequestsActivity to handle member requests.

## activity\_manage\_requests

The activity\_manage\_requests.xml file defines the user interface for the ManageRequestsActivity. It provides a list of item requests from household members and enables the Head of Household to approve or deny these requests.

Key UI Components

* RecyclerView requestsRecyclerView: Displays the list of item requests.
* ImageButton backButton: Allows users to navigate back to the previous activity.
* TextView introText: Displays a brief introduction or instructions for managing requests.

## activity\_register

The activity\_register.xml file defines the user interface for the RegisterActivity. It provides input fields for users to create an account, including optional fields for joining an existing household.

Key UI Components

* EditText usernameEditText: Captures the user's chosen username.
* EditText passwordEditText: Captures the user's chosen password.
* EditText householdIDEditText: Allows users to enter a Household ID if joining an existing household.
* Button registerButton: Submits the registration form.
* ImageView splashImage: Displays the app's logo to enhance branding.

## activity\_shopping

The activity\_shopping.xml file defines the user interface for the ShoppingActivity. It displays the household's shopping list and provides options to mark items as purchased, add new items, or navigate back to the inventory.

Key UI Components

* ListView shoppingListView: Displays the list of shopping items.
* ImageButton addItemButton: Allows users to navigate to the EditItemActivity for adding new items.
* Button markAsPurchasedButton: Marks selected items in the shopping list as purchased.
* ImageButton backToInventoryButton: Navigates back to the InventoryActivity.

## activity\_splash

The activity\_splash.xml file defines the user interface for the SplashActivity. It provides a simple splash screen that displays the app's logo during the initial loading phase.

Key UI Components

* ImageView logoImage: Displays the app's logo at the center of the screen.

## household\_list\_layout

The household\_list\_layout.xml file defines the layout for individual rows in the ListView displayed in the ManageHouseholdActivity. Each row represents a household member, showing their username and permissions.

Key UI Components

- TextView usernameTextView: Displays the username of the household member.

- Switch permissionSwitch: Allows the Head of Household to toggle permissions for the member.

## inventory\_list\_layout

The inventory\_list\_layout.xml file defines the layout for individual rows in the ListView displayed in the InventoryActivity. Each row represents an inventory item, showing details such as name, quantity, and unit, and providing buttons for editing or adding items to the shopping list.

Key UI Components

* TextView itemNameTextView: Displays the name of the inventory item.
* TextView itemQuantityTextView: Displays the quantity of the item.
* TextView itemUnitTextView: Displays the unit of measurement for the item.
* ImageButton editButton: Allows users to navigate to the EditItemActivity to edit the item.
* ImageButton addToShoppingListButton: Adds the item to the shopping list.

## requests\_list\_layout

The requests\_list\_layout.xml file defines the layout for individual rows in the RecyclerView displayed in the ManageRequestsActivity. Each row represents a request from a household member, showing details such as item name, quantity, and unit.

Key UI Components

* TextView itemNameTextView: Displays the name of the requested item.
* TextView itemQuantityTextView: Displays the quantity requested.
* TextView itemUnitTextView: Displays the unit of measurement for the requested item.

## shopping\_list\_layout

The shopping\_list\_layout.xml file defines the layout for individual rows in the ListView displayed in the ShoppingActivity. Each row represents a shopping list item, showing details such as name, quantity, and unit, and providing a checkbox for marking the item as purchased.

Key UI Components:

- CheckBox purchasedCheckBox: Allows users to mark the item as purchased.

- TextView itemNameTextView: Displays the name of the shopping list item.

- TextView itemQuantityTextView: Displays the quantity of the item.

- TextView itemUnitTextView: Displays the unit of measurement for the item.