Team: Three Blind Mice

Members: Andrew Chang (afchang@andrew),
Tanay Gavankar (tgavanka@andrew),
Steven Kang (sskang@andrew)

Architecture Design

Business Context:

- Client: Michael Bigrigg
- Product: The client wants a product to track and record inventory of items (food) in the kitchen for blind users. The product will be used to track what items a user has thrown away. This means that a shopping list should be generated quickly and on demand, without needing to take inventory of the kitchen.
- Deployment: Expected to be used on a dedicated computer for inventory tracking
- The client would also like to be able to have a product for non blind users, however this is not the primary objective

Architectural Drivers

1) Technical Constraints
    - Users are visually impaired.
    - Internet access maybe limited.
    - Client prefers Java as the primary language.
    - Must operate with a barcode scanners to scan in UPC codes
    - Need a lightweight database that does not require set up and able to maintain up to hundreds of thousands of data entries
    - Minimal library dependency to allow platform independency
2) Quality Attributes - ordered by importance (first is most important)
    - Usability
        - Our core user base is blind, so we need to build an application that supports accessibility standards.
        - Every string that we present to the user should be recognized by any screen reader.
    - Extensibility
        - We want to be able to easily add new features to the application as development moves forward
        - The cost of developing a new feature should take no more than 16 hours of development time. (This does not include research or planning or extra architecting, only development time. If it does, it should be broken up into smaller features.)
    - Portability
        - Our software should not depend on the operating system or installed software packages
3) High level Functional Requirements
    - Scan an item and record that it was thrown away

- Generate a shopping list of items.
- Be able to export the shopping list to a file.
- Be able to email the shopping list to any recipient
- Support console UI and GUI.

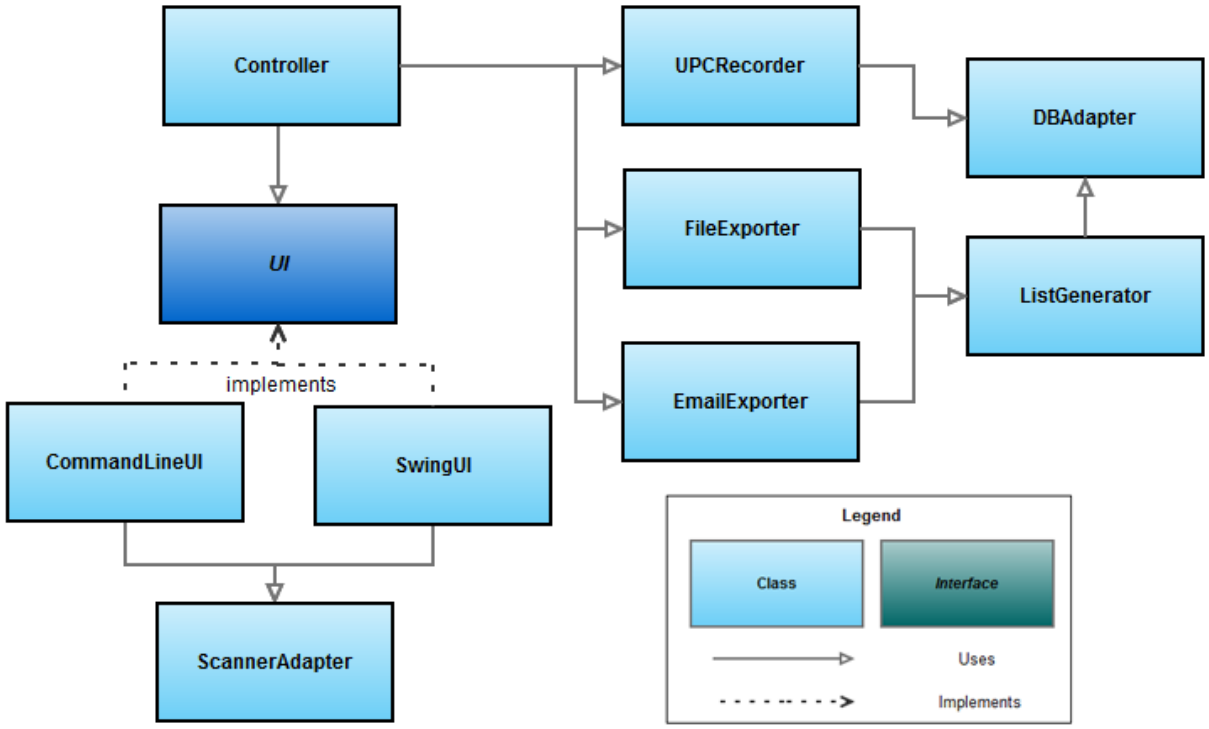
Assumptions

- The computer running the software is dedicated for the program.
- The computer may not have constant internet connection, but it will have occasional internet access.
- The user is comfortable with using a computer.
- The barcode scanner is sending the scanned UPC as keyboard input.

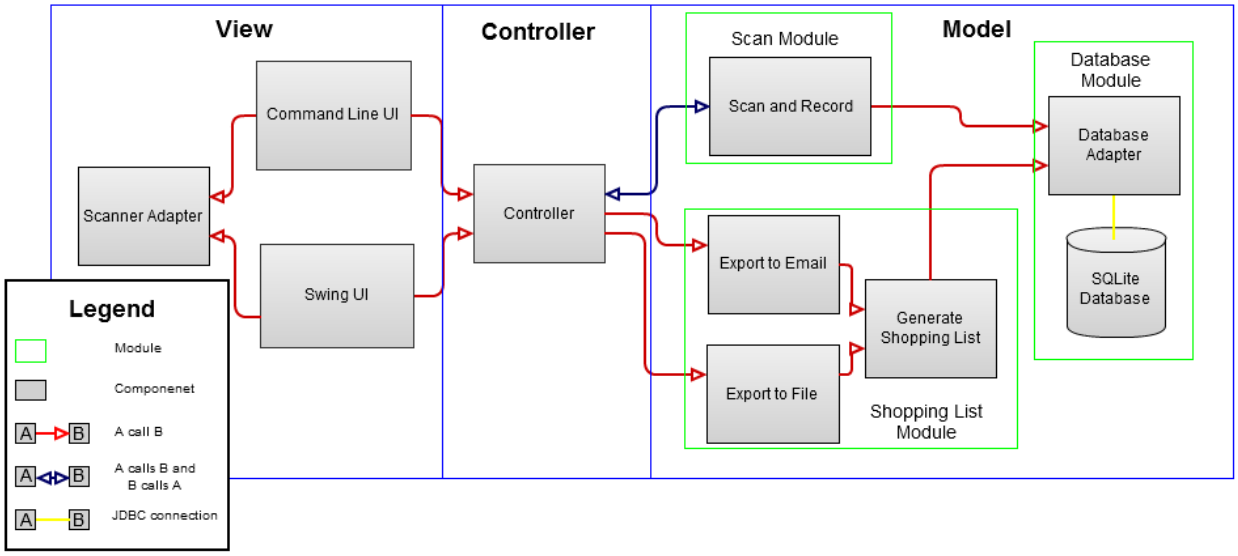
Justification

These diagrams support our three top quality attributes. First, we prioritized usability with a common UI interface. Thus, we can build custom user interfaces that support the accessibility requirements. This way, we allow the system to be accessible to both blind and non-blind users with minimal changes to the system. We also followed the Model-View-Controller pattern to allow maximum flexibility and extensibility. We can simply implement new classes for the model to add new functionalities. The combination of Java and SQLite provides platform independence, thus guaranteeing maximum portability.

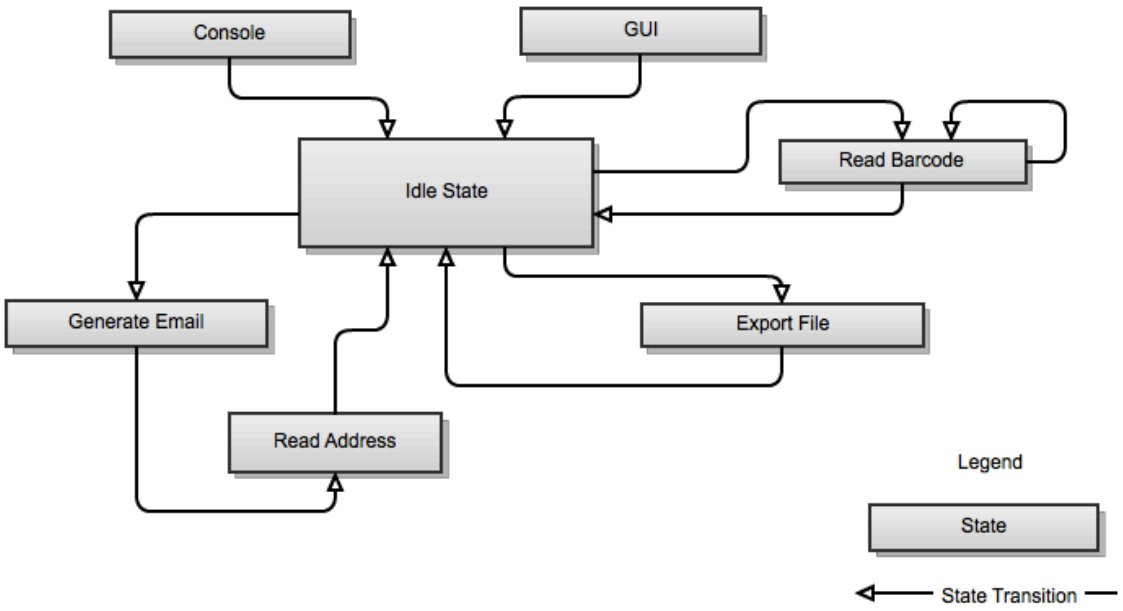
Class Diagram



Dynamic View



KIT Workflow Diagram



The workflow diagram starts at either the console or the GUI - that will depend on which one the user wants to use (multiple launch shortcuts). After starting the application, the system is in an idle state, waiting for user input. If the user wants to start scanning items, then we enter the 'Read Barcode' loop. This loop will read UPC codes from the barcode scanner and store them in the database. Then loop until the user decides to stop. Futhermore, the user can sent commands to export a shopping list to a file or generate a shopping list to email. All commands and paths eventually return to the idle state to receive more user input.