

GitHub Username: [jheske](#)

# Deckbot

## NOTES

- I haven't found any use yet for a second Google Play Service. Any suggestions would be appreciated.
- I did not include every single screen shot. Some alternative layouts are not shown, but all will be shown in final app. For example, image RecyclerViews will be multi-column grids on tablets.
- Magic the Gathering is a **\*\*really\*\*** complicated game with a large and demanding fan base! This app is just the tip of the iceberg as far as what users will expect. There are many features I could have included in both the AI library and the UI, particularly deck and card stats for analysis, but since this is the first version, I kept it as simple as possible.

## Background

Magic the Gathering, or simply "Magic" or "MtG", is a trading card game published in 1993 by Wizards of the Coast and enjoys as many as twenty million players worldwide. The game requires two or more players, each of whom uses a deck of sixty or more printed or virtual cards. From Wikipedia:

*Magic is currently in production and new cards are released on a regular basis through expansion sets. An organized tournament system played at an international level and a worldwide community of professional Magic players has developed, as well as a substantial secondary market for Magic cards. Certain Magic cards can be valuable due to their rarity and utility in gameplay. Prices range from a few cents to thousands of dollars.*

The universe of Magic cards currently comprises approximately 30,000 Magic cards, around 16,000 of which are unique (have the same name and playability as other cards but show different artwork on the card face). Card prices range from less than ten cents to as much as \$15,000! Players include everything from young hobbyists to professional players, a select few of whom actually make their living touring and competing in MtG tournaments.

## Problem

Players of all levels are naturally interested in building the strongest and most effective decks for their chosen application within an affordable price range. While there are apps to help players search for cards and build analyze decks, none of the apps is able to optimize card selection

based on cost constraints, making it difficult for aficionados of all levels to make the most of their often limited budgets. Deckbot proposes to begin solving that problem.

## Proposed Solution

Design an app called Deckbot that will allow a user to specify preferences based on several MtG-relevant criteria, including *color*, *flavor*, and maximum total deck *cost*, pass preferences to a library which implements an AI engine, called *Deckbot-AI*, which will return a deck conforming to user preferences, whose total cost is within the user's budget and is legally playable under rules defined by Wizards of the Coast.

## Intended User

Professional and amateur Magic the Gathering players

## Features

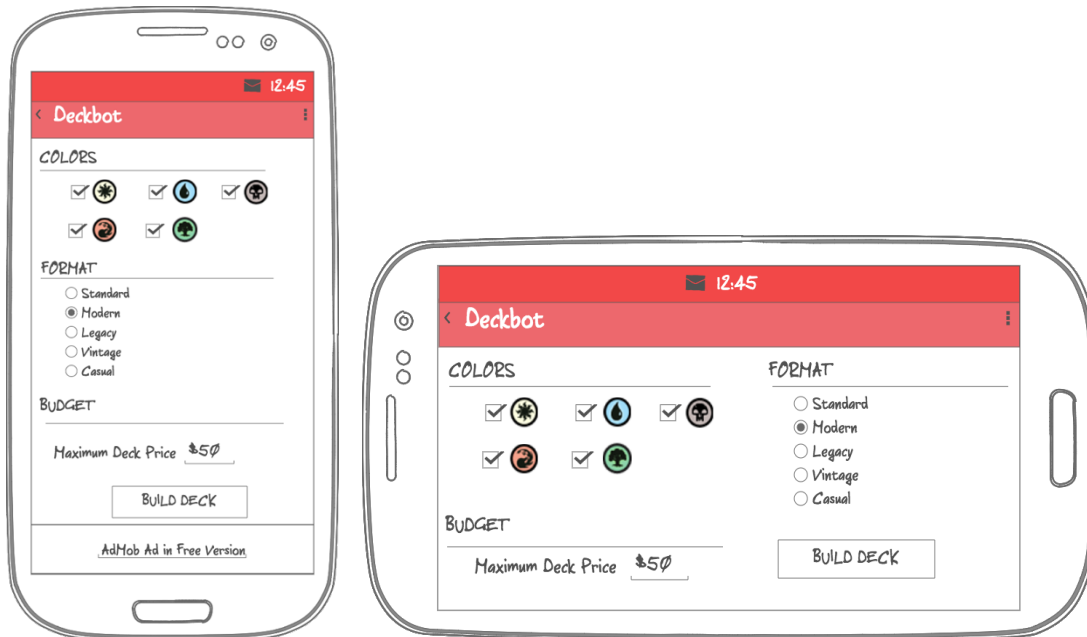
- Download/update entire universe of MtG cards and most recent fair prices
- Based on user's color and format selections, DeckbotAI generates a legal deck within specified budget
- Charts and metrics for analyze resulting deck
- View, analyze and manage previously-created decks
- Build as many decks as you want
- Browse entire universe of MtG sets

## User Interface Mocks

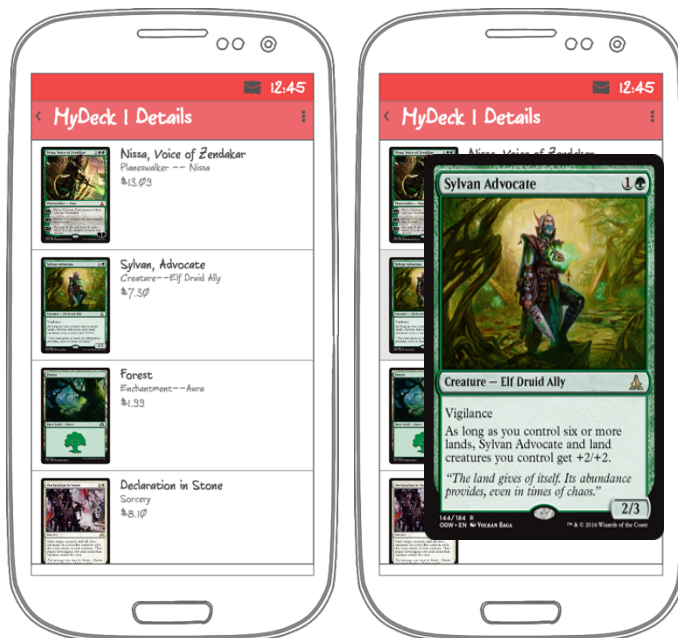
### Navigation



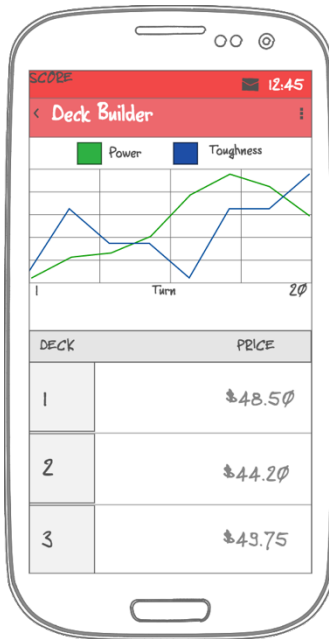
## Home



## Deck Builder Results/Deck Details



## My Decks



## Sets



**Sets**

Update Sets

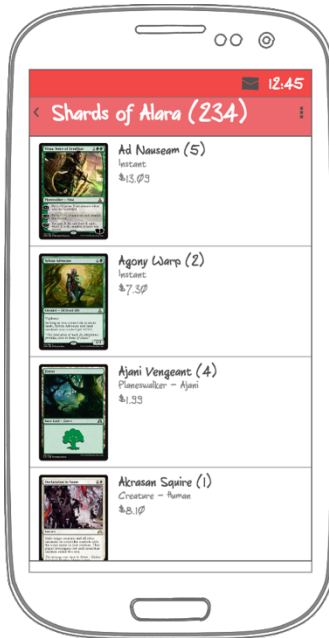
ALA	Shards of Alara Oct 3, 2008
ALL	Alliances Jun 6, 1996
APC	Apocalypse Jun 4, 2001
ARB	Alara Reborn Apr 30, 2009
ARC	Archenemy Jun 18, 2010
	Arabian Nights

**Sets**

Update Sets

ALA	Shards of Alara Oct 3, 2008	 Ad Nauseam (5) Instant \$13.09
ALL	Alliances Jun 6, 1996	 Agony Warp (2) Instant \$7.30
	Apocalypse	

## Set Details



## Data and Persistence

Deckbot uses a `ContentProvider` to access a SQLite database. Use `android-contentprovider-generator` to auto-generate `ContentProvider` code. The utility, developed by **Benoit Lubek**, can be found at <https://github.com/BoD/android-contentprovider-generator>.

## Card and Price Updates

- Users will rarely need to update cards, just a few times a year when Wizards releases new sets of cards
- Prices should be updated daily for most accurate results.

## Libraries

- `appcompat`, `recyclerview`, `cardview`, `design`, `gridlayout`, etc.
- `com.google.android.gms:play-services-ads` – *Free version only*
- `com.squareup.phrase:phrase` – *Android string resource templating*
- `com.facebook.stetho` – *View database using Chrome browser*
- `com.jakewharton:butterknife` – *Bind Android views and callbacks to fields and methods*  
*// For nav\_header.xml*
- `de.hdodenhof:circleimageview` – *NavigationDrawer header*
- `com.squareup.retrofit2:retrofit` – *Interface API services*

- `com.squareup.retrofit2:converter-jackson` – *JSON parser/converter*
- `com.squareup.okhttp3:logging-interceptor` – *Retrofit debugging*
- `com.squareup.picasso:Picasso` – *Image downloading*
- `com.github.BlaCkCaT27:CurrencyEditText` – *Currency-filtered EditText*
- `com.github.PhilJay:MPAndroidChart` – *Graphs and charts*
- `com.h6ah4i.android.widget.advrecyclerview:advrecyclerview`
- `com.heske.deckbot-ai` – *proprietary deck building library written in C and accessed via JNI*

## Tasks

### Task 1: Project Setup

- Configure libraries and flavors in `build.gradle`
- Create JNI folder structure
- Add `deckbot-ai` C files to JNI directory, or alternatively, add `.so` file to library folder
- Create Free and Paid src folders
- Create database structure json files for use with `android-contentprovider-generator`
- Use `android-contentprovider-generator` to generate `ContentProvider` files. See <https://github.com/BoD/android-contentprovider-generator> for instructions.
- Gather resources, including MtG-specific graphics and Material launcher and menu icons for `mdpi`, `hdpi`, `xhdp`, and `xxhdp` devices

### Task 2: Create Theme

- Add colors to `colors.xml`
- Add styles to `styles.xml`

### Task 3: Implement UI Resources

Provide portrait and landscape layouts for phone and tablet designs depending on best use of space for various orientations and screen sizes.

- Build layouts for `MainActivity` and `MainActivityFragment`, integrating MtG-specific icons
  - Colors checkboxes: Black, White, Blue, Green, Red
  - Format radio group: Standard, Modern, Legacy, Vintage, Casual
  - Budget `EditText`
  - `BuildDeck` button
- Build layouts for Navigation drawer layout, header, and menu
  - Menu Items: Home, Decks, Cards, About, Settings
- Build layouts for `MyDecks`

- RecyclerView listing Decks
- Graph showing Power and Toughness for selected Deck
- Build layouts for DeckAnalysis screen (analyze a single deck and its cards)
  - RecyclerView listing Cards
  - TextViews showing selected deck's details and metrics
  - If time permits integrate graph and chart widgets

#### **Task 4: Create JNI Wrapper**

- Implement wrappers for library calls

#### **Task 5: Integrate data from MtgJson.com and MtgPrice.com**

- Implement MtgApplication class and MtgJsonApiService and MtgPriceApiService
- Implement data model POJOs for cards, decks, sets, and auxiliary components such as legalities, colors, etc., and annotate for Jackson parsing
- Implement DatabaseUtils class for adding parsed data to the database

#### **Task 6: Implement IntentServices for Retrieving Cards and Prices**

- Implement MtgJsonDataService to retrieve cards from MtgJson.com
- Implement MtgPriceDataService to retrieve prices immediately from MtgPrice.com
- Services will provide background notifications with ProgressBars

#### **Task 7: MainActivity and MainActivityFragment**

- Hook up widgets for user selections
- Declare JNI wrapper functions
- Hook up Build Deck button to call DeckbotAI library in a thread and launch DeckBuilderResultsActivity on thread completion

#### **Task 8: MyDecksActivity and MyDecksActivityFragment**

- Populate RecyclerView with list of cards in latest deck
- Select first deck and display its graph results

#### **Task 9: MyDeckActivity and MyDeckActivityFragment**

- Populate RecyclerView with card images and stats
- Each card shows stats returned from AI library
- Long press shows larger card image

#### **Task 10: MtgSetsActivity and MtgSetsActivityFragment**

- Populate RecyclerView with list of all Sets from the database

- Implement Update Sets to call IntentServices to update sets and prices in the background

### **Task 11: MtgSetActivity and MtgSetActivityFragment**

- Populate RecyclerView with all cards in a set
- Long press shows larger card image

### **Task 12: Integrate Google PlayServices AdMob**

- Add AdMob into MainActivity

### **Task 13: Implement Home Page Widget**

### **Task 14: Corner Cases**

- Inform user if AI cannot create any decks based on user criteria
- On initial setup, prompt user to download card and price data. Subsequent updates are required only a few times per year and are done on Sets screen.
  - Display msg on UI if not connected to internet

### **Task 15: Sign App for Release**