**Capstone Project Phase 1:**

**Team:** Marco Lauguico, Peter Tang, Kevin Do

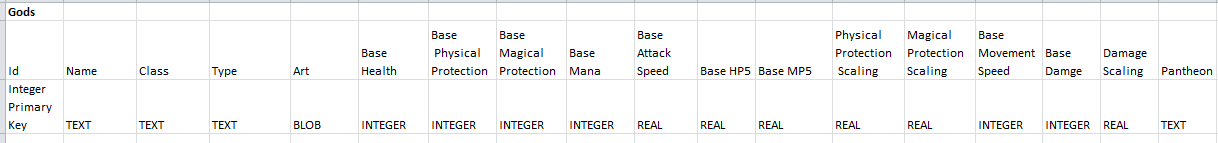
**Project Name:** Smite Item Builder

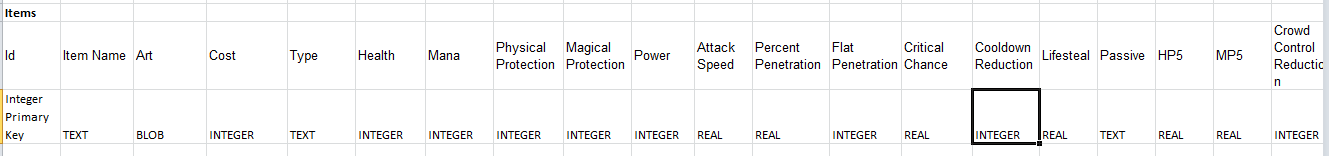
**Purpose:** The game Smite has a lot of characters with different stats, and with that they are able to build 6 items. The game has hundreds of items to choose through, and this application will allow the user to test out different build combinations to best suit what they want. The app will have everything from god stats, item stats, item descriptions, lore and much more.

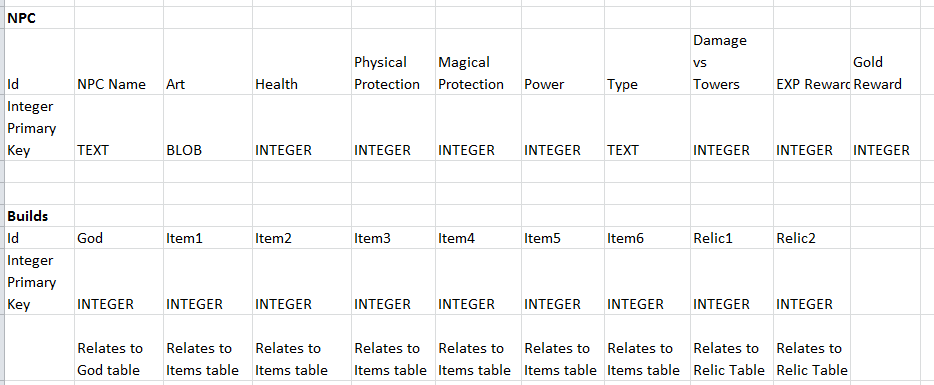
**Audience:** Users who play the game Smite whether it be new players or professionals

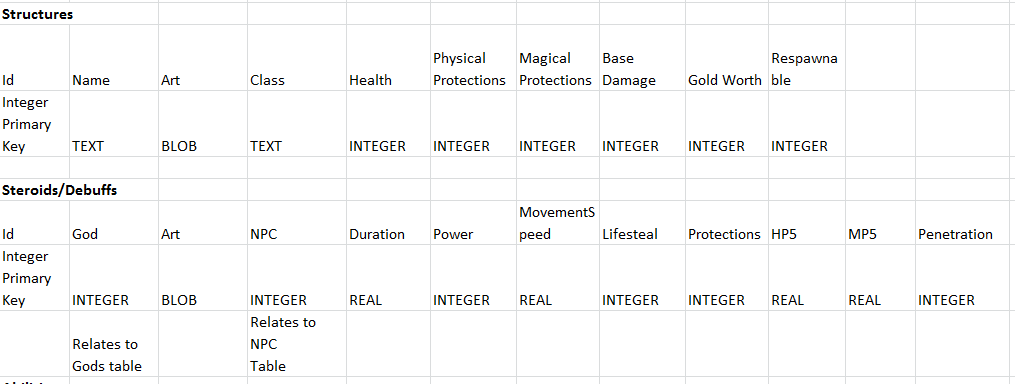
**Non-Profit:** May release it, need intent of teammates.

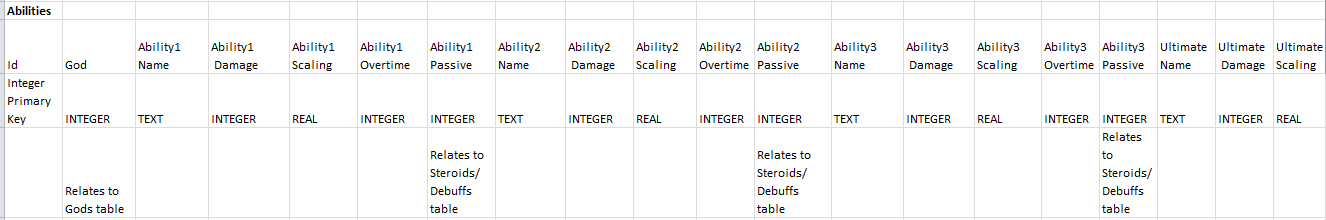
**Database Listing:**

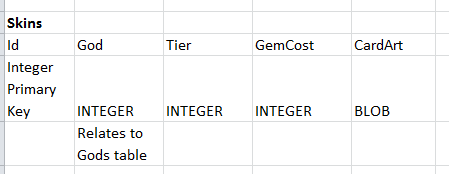


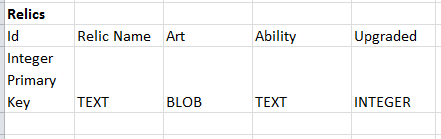




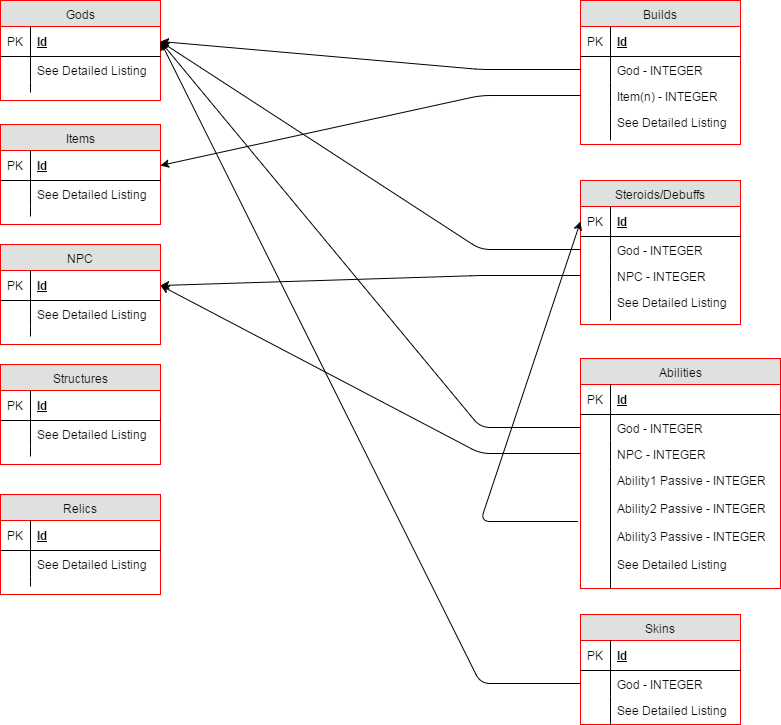




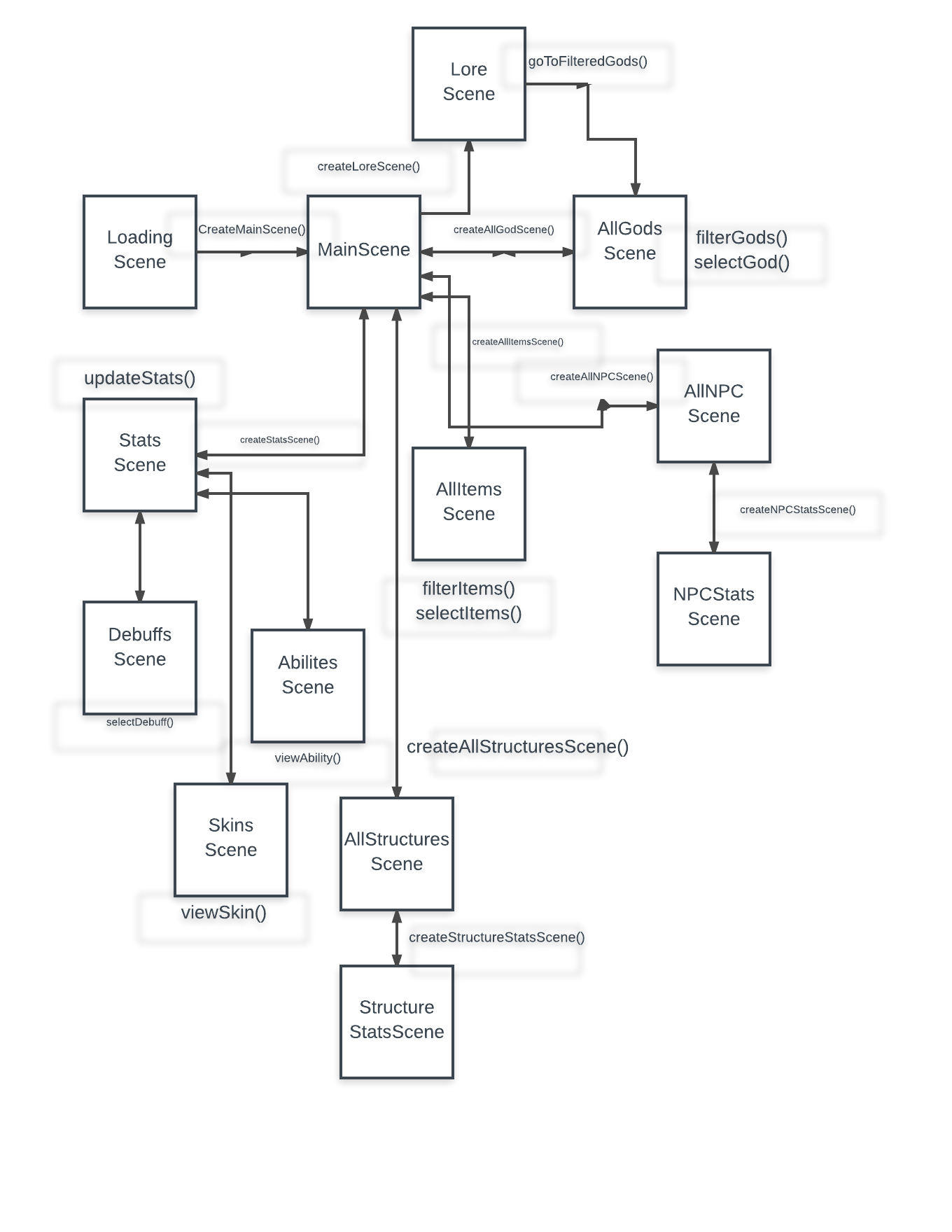




**Database Schema:**

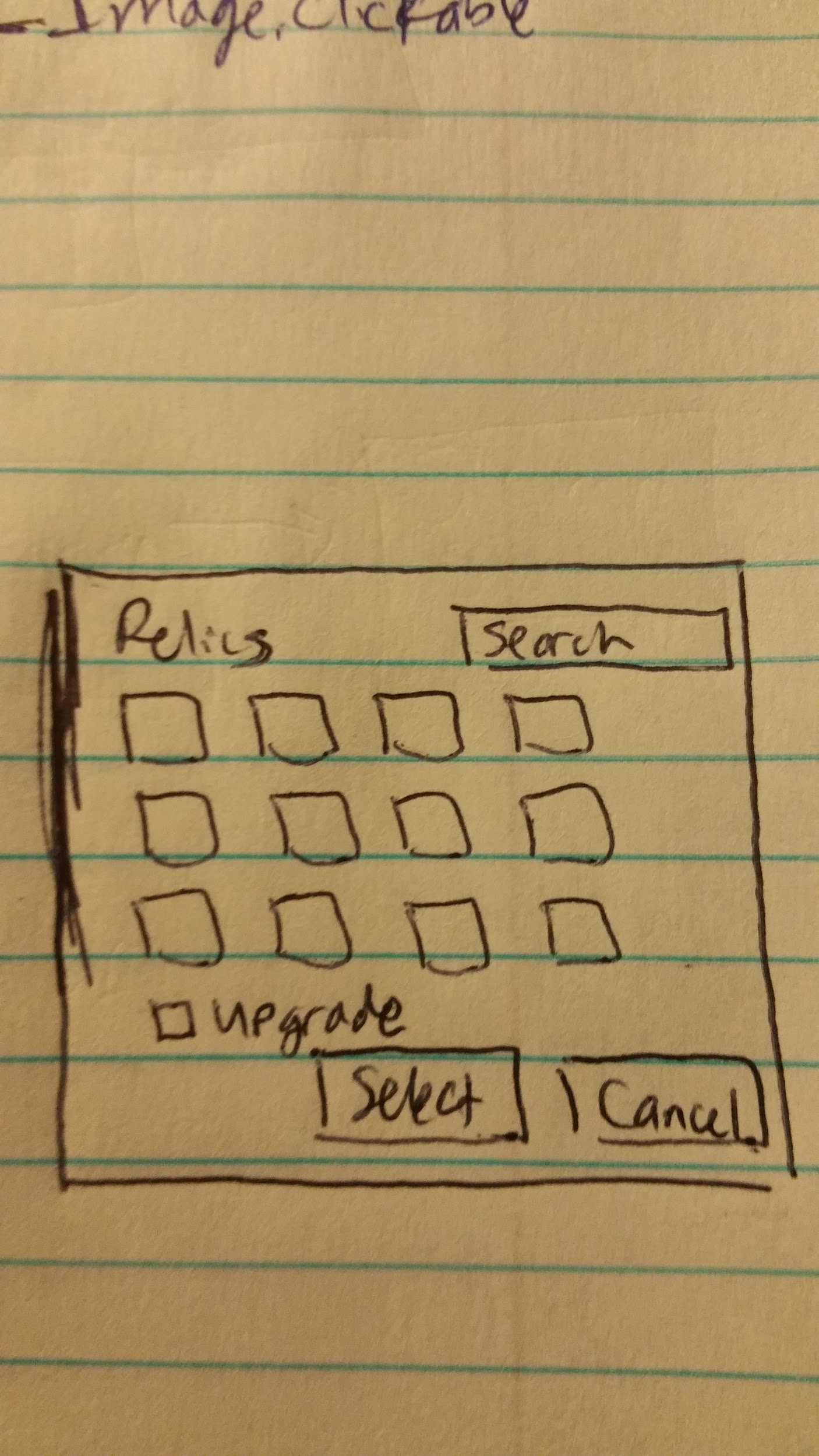


**Scene Flow**

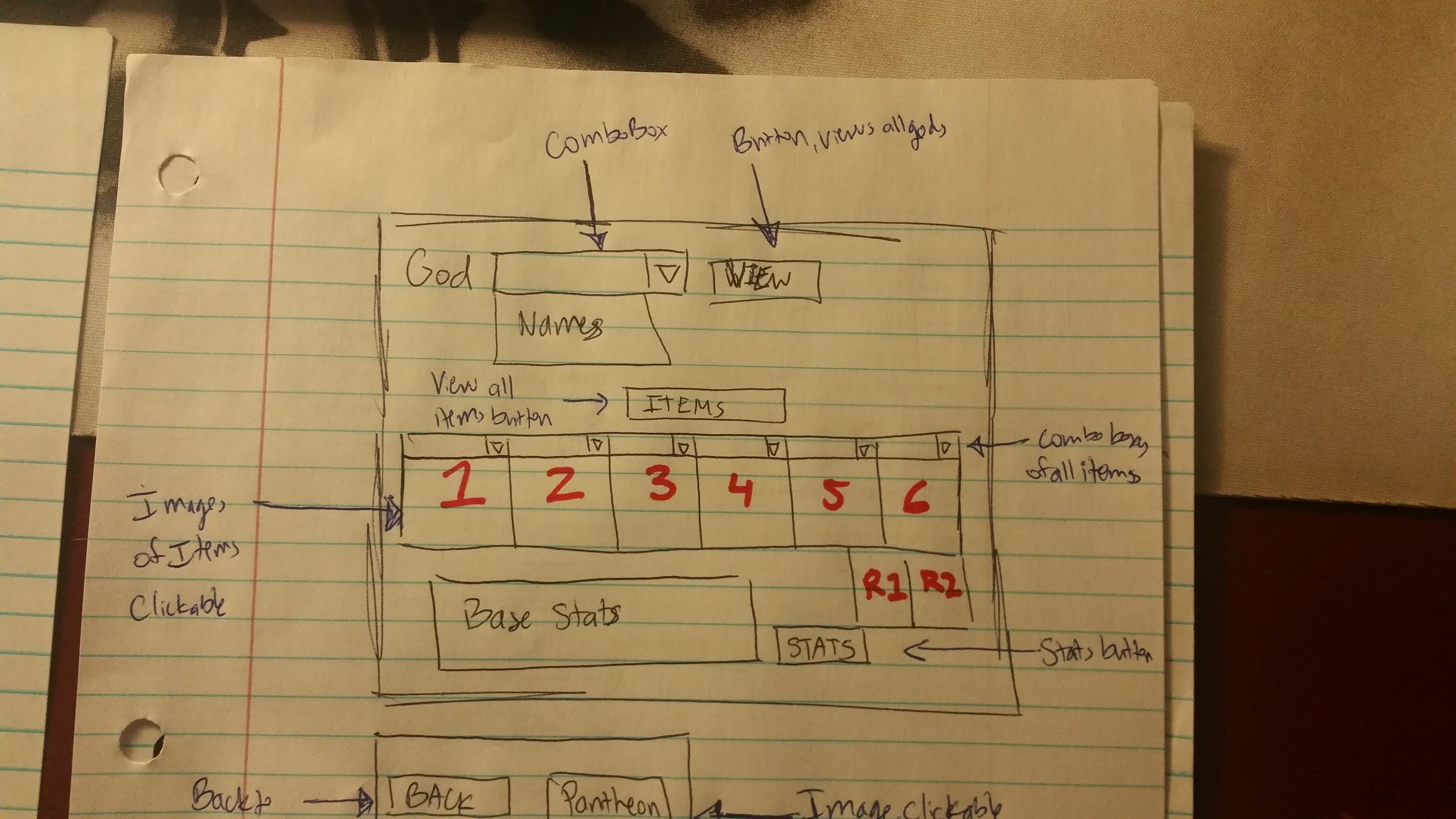


**Examples of Scenes:**

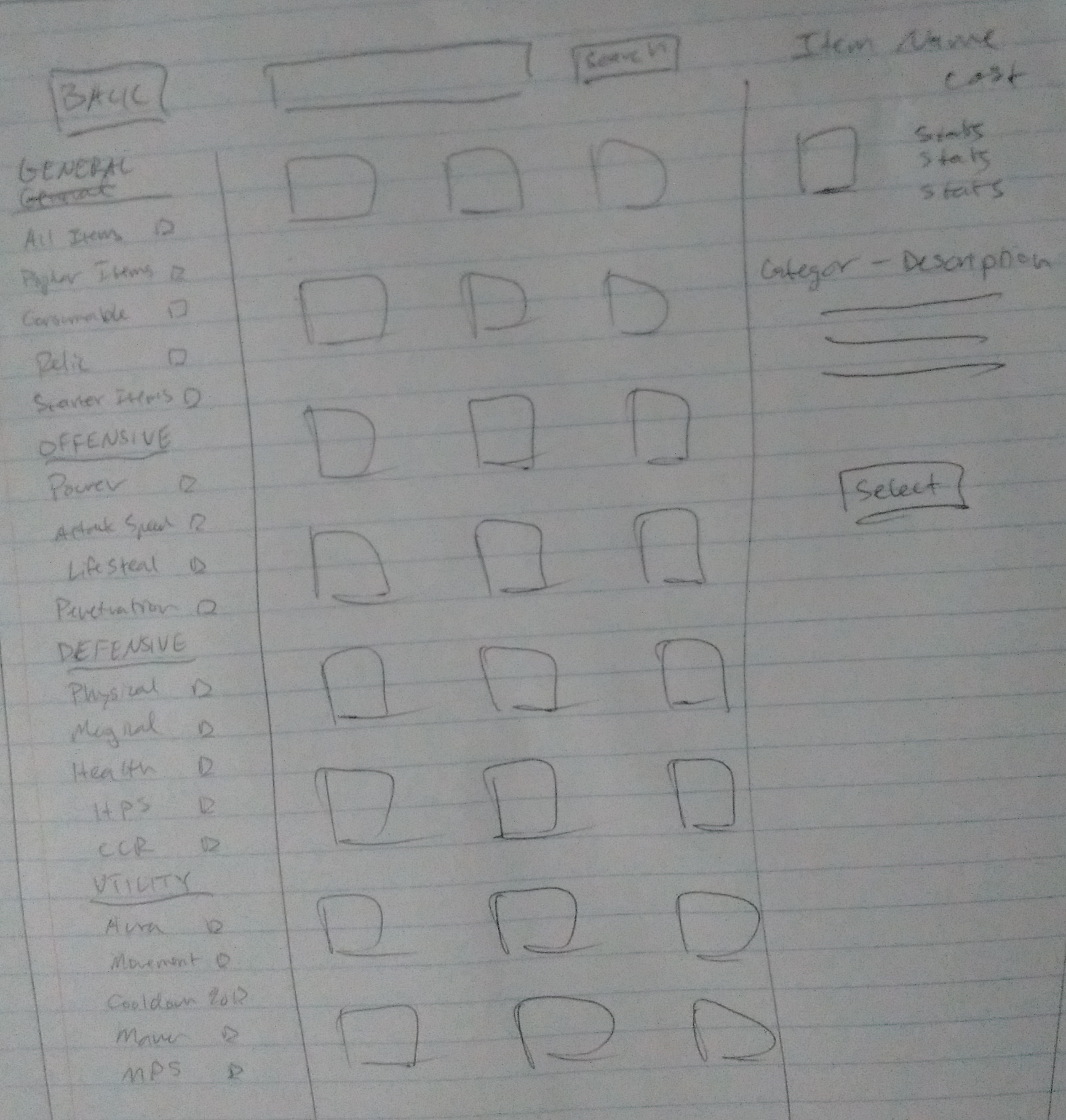
Relics Scene ->shows all relics, can choose to add to build



Main Screen -> View all gods, view god stats, view all items, view lore

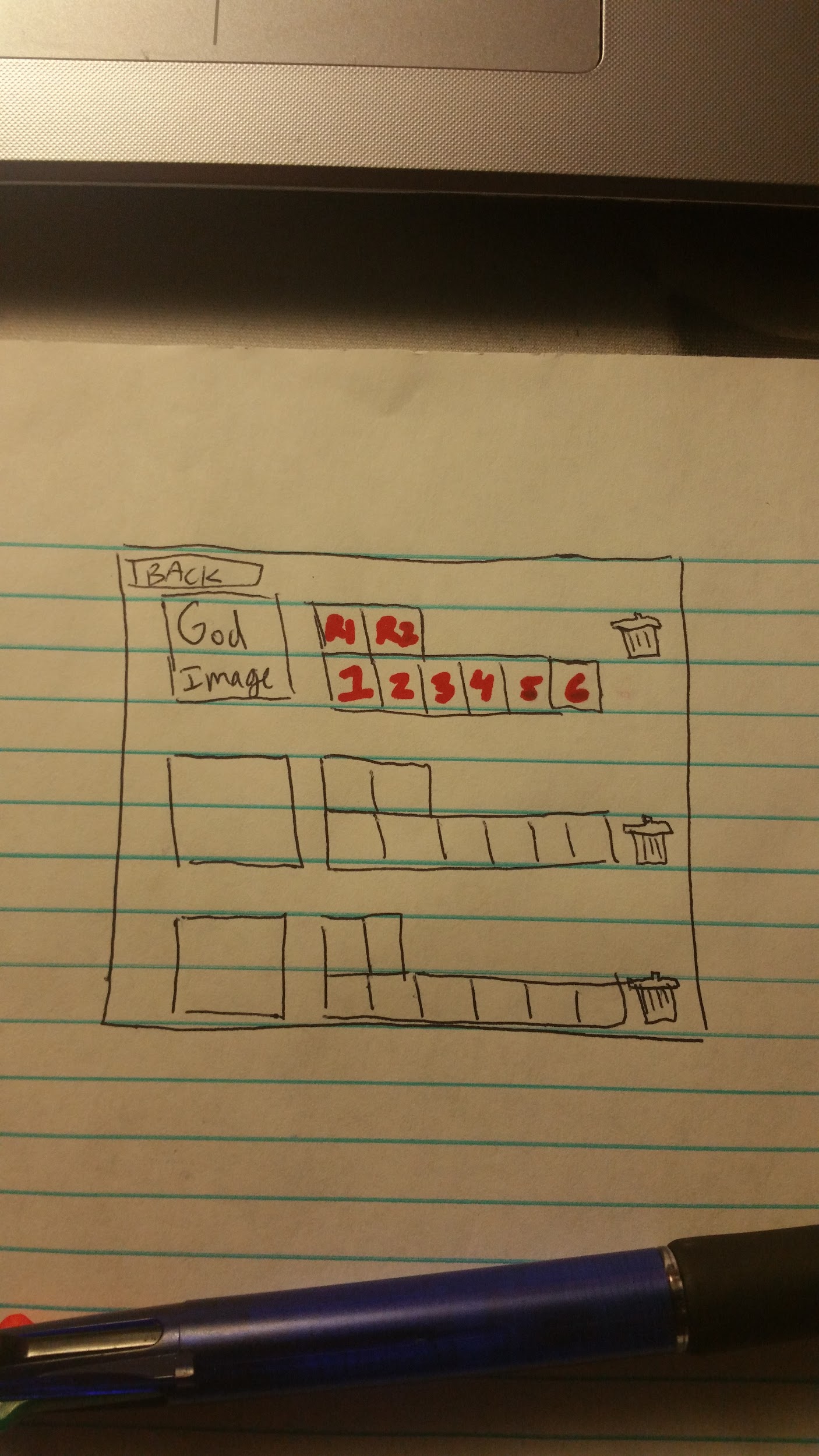


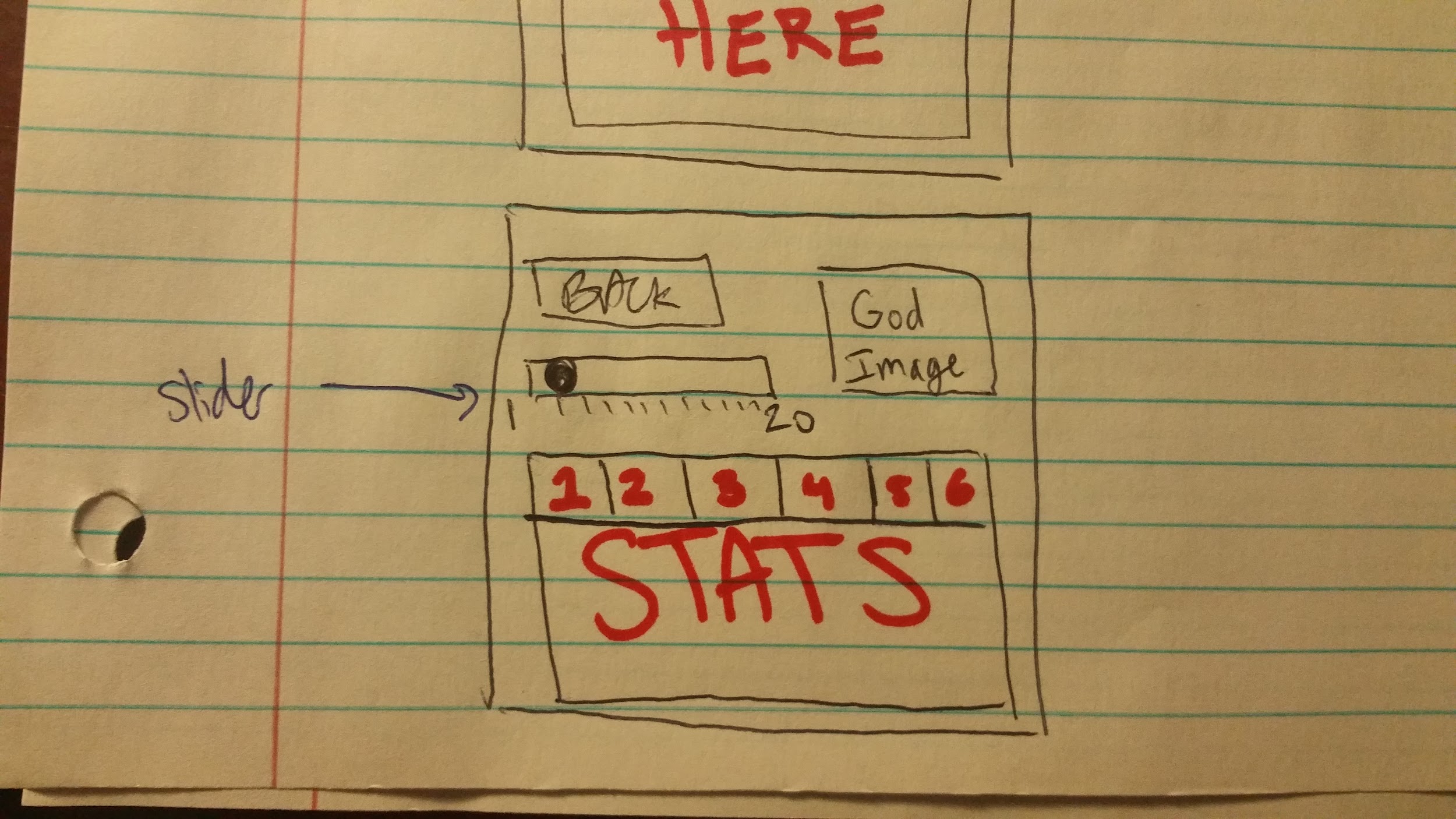
All Items Page -> Filter items, Checkboxes for different types of items, Back to main screen

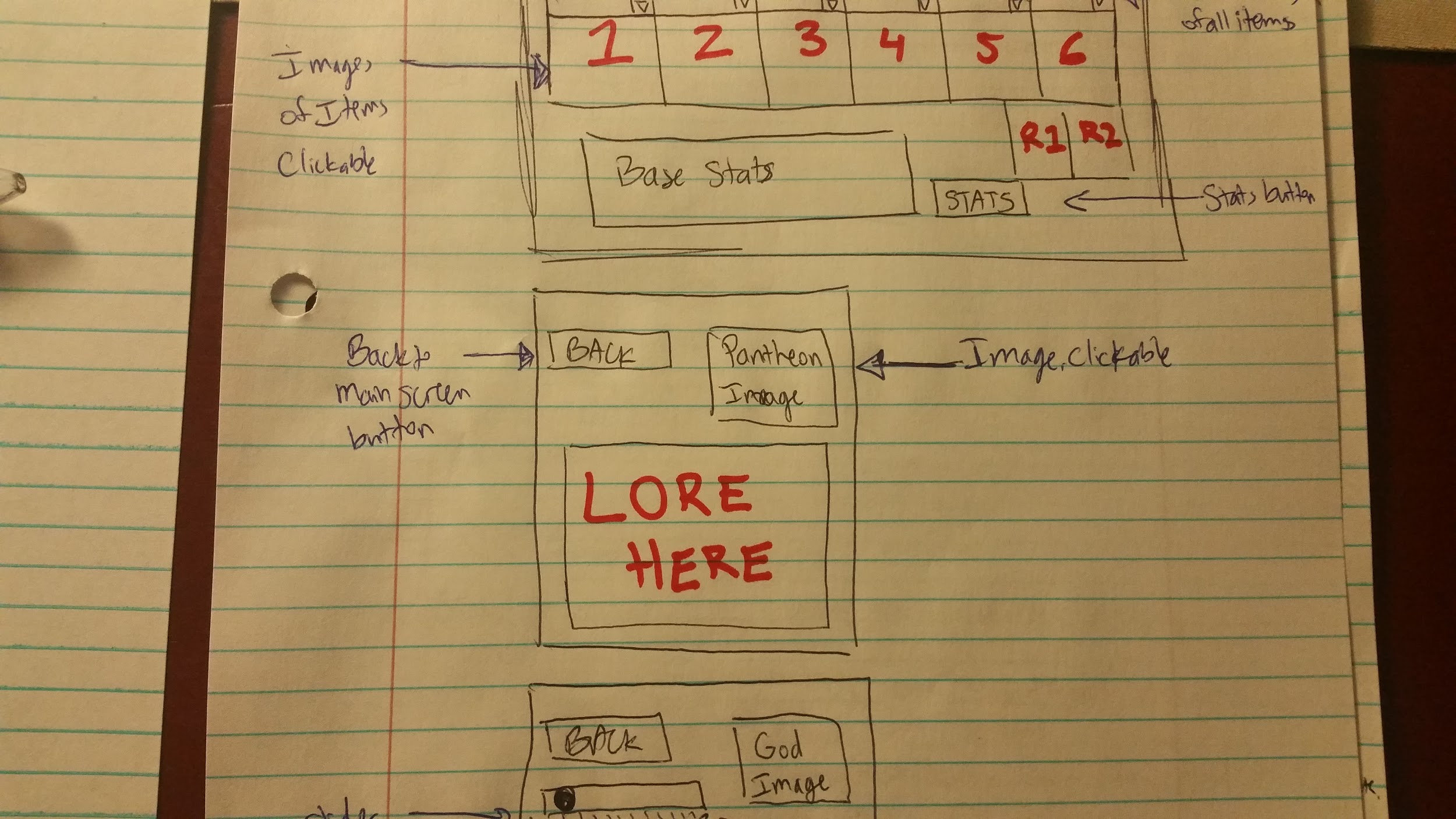
 Essentially a close adaptation of -->



View Saved Builds -> load saved builds, delete builds, search builds, Back to main screen



Stats page -> Update stats based on Lv (from slider), Back to main screen

Lore page -> clickable pantheon logo shows all gods of that Pantheon, Back to main screen

**Activities**

* viewAllGods() - this is an activity that will bring the user to a list of all god icons on which they can press the god’s icon and once they did, it’ll bring them to a scene where the God’s attributes are displayed. This list comes from the model (database), creates tables from database if not already done so
* viewAllItems() - this is an activity that will prompt the user to a list of all item icons on which they can press and once they did, it’ll prompt them to a scene where the item’s description and item effects are displayed. This list comes from the model (database), creates tables from database if not already done so
* viewGod() - this is a scene where it displays a God’s attributes. \*Name, Description(lore), Stats, and Abilities. Pulls the info from the model (database), creates tables from database if not already done so
* viewItem() - this is a scene where it displays an item’s description and effects. Pulls data from the model (database), creates tables form database if not already done so
* filterItems() - filters out list of all items per checkbox parameters
* searchItem() - searches subsequence in item name with all filters applied
* viewAllBuilds() - this is an activity inside the viewGod() scene where if is active will bring the user to the scene where all the list of all possible builds for that certain God is displayed.
* addBuildScene() - this is an activity where if a user activate it will bring the user to a scene where they can name the build and etc and add it on the build database.
* addBuild() - this is an activity that creates a new record of a build for a God in the build database.
* deleteBuild() - this is an activity that removes a God build record from the build database.
* selectItem() - this selects the items, takes the user to the main screen, adds it to the build and updates all stats accordingly
* selectGod() - this selects the god, takes the user to the main screen and updates all stats
* selectRelic() - this selects the relic, takes the user to the main screen, and adds it to the build
* editBuild() - this is an activity that will prompt the user to the addBuildScene() and will load the previous values for the build and the user can edit them and updates it on the build database.
* viewBuildScene() - this is an activity where if activated will bring the user to a scene where the stats and the item build of a God are shown and a slider from (1-25) that will show the stats on a certain level.
* sliderStatHandler() - this activity is connected to the viewBuildScene() where there is a slider from 1-25 and will update the displayed stats from a certain God according to the level.

|  |  |  |
| --- | --- | --- |
| Model | Controller | View |
| All Database tables above   * All entity class member variables mirror its corresponding database attributes   **Gods.java**   * mId * mName * mClass * mType * mArt * mStats * mBuilds (An arraylist/dynamic array of build class) * mPantheon   **Items.java**   * mId * mName * mArt * mCost * mType * mStats   **Relics.java**   * mId * mName * mArt * mAbility * mIsUpgraded   **Structures.java**   * mId * mName * mArt * mClass * mHealth * mPhysicalProt * mMagicalProt * mBaseDamage * mGoldWorth * mIsRespawnable   **Npc.java**   * mId * mName * mArt * mStats * mExp Gain * mGold Gain   **SteroidsDebuffs.java**   * mId * mGod * mArt * mNPC * mDuration * mPower * mMoveSpeed * mLifesteal * mProtection * mHP5 * mMP5 * mPenetration   **Builds.java**   * Id * God * inventory   **Abilities.java**   * See above DB description | * Singleton Pattern * So far planning to go with one controller * Examples of methods within the controller to process and return result to the View * viewAllGods() * viewGod() * searchGod() * filterGods() * viewAllItems() * filterItems() * viewItem() * searchItem() * selectItem() * viewBuild() * sliderStatHandler() * searchBuild() * viewAllRelics() * selectRelic() * filterRelics() * upgradeRelic() * addBuild() * editBuild() * deleteBuild() * viewLore() * Remaining scenes follow a template to view all items of the game’s attribute with the ability to filter through the list and show the filtered results | * JavaFX application * Scenes listed by scene name   Splash/Title/Loading Scene  AllGodsScene   * View with all gods selection   AllItemsScene   * View with icons of all items   StatsScene   * View of God’s stats   RelicsScene   * View with all relics   SavedBuildsScene   * View with editable build   LoreScene   * View with God’s lore   AllNPCScene  NPCStatsScene  DebuffScene  AbiltitiesScene  AllStructuresScene  StructureStatsScene  SkinsScene  etc. |