# Project Summary

As recommend, I am going to use Django to build and issue tracker and product catalog. This program will be organized into the following modules.

1. Home page – that provides a welcome page to Harry Tools and provides a list of our anticipated projects. On this page will be a menu button to the “issues” page.
2. Issues page – will be heart of the project and fashioned to functionally resemble Github’s issue pages. Users will be able to vote which features and issues an user would prefer to see fixed.
3. Product Roadmap page – will provide a summary of on-going features and issues requests being worked on and sorted by product. Also included on this page will be graphs showing the votes for top features and issues.
4. Product Page - A product page, which is reached through the home page, will include products to be purchase. From this page, a user may add products to a shopping cart for purchase.

# Home Page Summary

The home page will provide generalized marketing information. It’s goal is to be the gateway to other features and basically to look nice and provide a high-level overview our services. On that home page, there will a top-level menu for:

|  |  |
| --- | --- |
| Log in | Allow users to log in or register as a user. The system will have an operating user authorization and registration system. |
| Products | Provides a web page that describes products and allows their purchase  (not products yet exist ) |
| Issues | Access to issue log pages |
| Shopping Cart | Access to shopping carT for purchases. Requires user to log in |
| Help | Dummy help page link |

# Products

The product page will provide an overview of the products and enable the user to add various products to the cart. The products will be Harry Tools’ financial management tools and will be organized if they were subscriptions to a service i.e. add-ons to financial management portal.

Each product will have a standard price and can be added to a shopping cart. Before the user can make a purchase, they will be required to (a) log in or (b) register and then log in. The login and register pages will managed by the Authentication module.

Product RoadMap

The product roadmap page will show a:

* A schedule for products to be released
* A list of top features to be developed and bugs to be fixed
* A bar graph of the features and bug fixes by product and popularity

The page will be accessed to Product’s page and return to either home page or products page.

# Issues

The issues page will be organized like a blog with two sections, one for bugs and second for feature requests. To reach the issues page, a user must be log in and be registered.

Any registered user can add an issue or feature request. However, only that user can edit their post until the point, the issue moves from a pending status to being worked on. Also, only the admin can mark the issue as being worked on or complete. The user when they enter an issue they will able to:

* Identify the product and issues type (feature or bug)
* Describe the issue and update load screen-shot or pdf

Any user can add a comment, in blog type manner and vote for a change. All votes will be recorded a check will be performed to make sure an user hasn’t votetwice for a specific issue

Status will be displayed using badges for: Pending, Scheduled, Completed.

From issues page, user’s will be able to check on graphs that show the popularity of each of the top features. Also, each feature will have tally of votes. All delivered features and fixed bugs will be on a separate list of completed fixes and sorted by date. This list may be a separate page.

# Shopping Cart

The shopping cart will be standard fare list of items to be purchased. Once the purchase has been made, it will be cleared, and user returned to home page. The system will keep of track of orders by user.

# Admin

The admin panel will have a page for admin or delegate to moderate the issues, mark them as pending, being worked on, or completed. Items being worked will be assigned a target deliver date, that will show up on the issues page. Separate pages will be created just for:

* Orders
* Issues
* products

In all cases, aforementioned objects can be created, edited or deleted by the admin or admin delegate.

# Technologies

The technologies to be used in development include:

|  |  |
| --- | --- |
| Django / Python3 | Base framework. |
| Pillow | For file storage |
| Bootstrap 4 | For styling. Also, base.html will be created using a bootstrap template. Icons may used from various sources |
| PostGress DB | Underlying database |
| Heroku , AWS Buckets | For deployment |
| Stripe | For credit card processing |
| Graphs | Question – which do you prefer Bokeh, Pygal or plotly. Matplotlib looks ugly to me. I could also google charts, but why mess with pushing data to an JS array, where there good backend options. |

Anticipate Django Apps include:

|  |  |
| --- | --- |
| Authentication | User management |
| Admin Extra Management | Enables Admin to manage issues and orders, through separate dedicated admin pages. |
| Products | Includes product overview pages |
| Roadmap | Includes the pages logic of roadmap graphs. (Should this may be umped into product app?) |
| Issues | Includes pages and logic for issues log |
| Orders | Manages the purchasing process |

# Appendix A. Detail Pages

The paragraphs below include text for web pages and database fields and descriptions.

## HT Home Page

Below is potential text for home page.

*Harry Tools goal is to provide tools to help family offices and high net worth individuals tools to help evaluate and manage their investments. The Harry Tools is developing a growing series of apps that enables our customers to evaluate complex investments opportunities such as purchases of investment real estate, derivatives and brokerage accounts. It is our long-term goal to build an integrated asset manager that tracks your net worth across assets classes and platforms. As new tools are developed they will be added to our platform.*

## Product Cards

The product cards represent modules that can purchase. In each case, product will have the following attributes:

|  |  |
| --- | --- |
| product\_id | Int |
| product\_name | Short text |
| product\_description | Text field |
| Product\_price | Number |

The admin will be able products through the admin panel.

The product will have cards to describe the product, the products for the cards will be:

* 1. Real Estate models - multi- tenant
     1. Price $100 per year
     2. Ideal for analyzing properties with a handful tenants.
     3. Add graph and excel icons
  2. Real Estate models – mixed use and multifamily
     1. Price $100 per year
     2. Ideal for analyzing multi-family and retail properties
     3. Add graph and excel icons
  3. Real Estate models - campus
     1. Price $100 per year
     2. Ideal for analyzing complex, multi-building projects
     3. Add graph and excel icons
  4. Developer
     1. Price $75 per year
     2. Ideal for modeling condo development project
     3. Add graph and excel icons
  5. Structure note performance monitor
     1. Price $25 per year
     2. Ideal for tracking the performance of portfolio of structured notes
     3. Add graph

NOTE: since the products are not yet real, I am not going to create subscription manager to verify if user is allowed to use a specific app. But I recognize that systems such this application would require such a feature.

## Users

In order to use the issue tracking system or purchase a product, a user is required to log in. If an account does not exist, the user must register an unique user name and secure password.

The system will keep track of:

|  |  |
| --- | --- |
| User\_id |  |
| User name |  |
| Password |  |

We will use the native capabilities of Django.

## Orders

After an order is processed, the Order database will keep track of the purchase by user. The orders database will look like.

|  |  |
| --- | --- |
| Order\_id | Number |
| Order\_purchaser | User\_id of purchaser |
| Order\_item | Product id |
| Order\_date | Date of purchase |
| price\_paid | number |

Orders will able to be viewed in special admin page.

### Shopping Cart

The shopping cart will persist only for the user session. Such that, a context rather than database will be used. The shopping cart will contain list of cart objects. The cart object will have:

|  |  |
| --- | --- |
| Id | Number |
| Product being purchased | Product\_id |
| Product name | Product\_name |
| price | Product\_price |

## Issues

The issues log, will track of list of issues being tracked. An issue can be feature or bug. Initially the user will be able to set an issues as a bug or feature. But, if admin can edit this designation and fix it at specific value. The issues table will include:

|  |  |
| --- | --- |
| id | Unique id for the issues |
| type | This can set to either bug or feature. Default to bug |
| Author | User\_id of wrote the issue |
| Date | Of issue was created or updated |
| Title | Short tile for issues |
| Description | Detail text description of the issue |
| File | Document or image upload |
| Status | Can pending, being worked on, closed |
| Type\_edit | True = if user can redefine it as bug, feature |
| Desc\_edit | True = user can edit description |

A second model will be created for comments.

|  |  |
| --- | --- |
| Id | Unique id |
| Detail | Comment text area |
| date | Date of comment or last edit. |
| Author | User \_id of comment’s author. |

Methods,

* Retrieve open bugs
* Retrieve closed bugs
* Retrieve open features
* Retrieve closed feature

# Schema

