# Sprint 1 Documentation

Sprint 1 Goals: Create Kanban, Set up Virtual Environment, Create Django Project, Build (Models, URLS, Views, HTML Templates)

* Kanban: An agile method that helps individuals and teams manage their projects. Kanban’s can vary in section size however, they need to incorporate three main ideas. A **to do section,** what tasks are needed for that sprint, a **in progress section,** tasks that are being worked on, a **completed section**, were the task in progress is complete and works. In my git hub you can find a good idea of how one will look (<https://github.com/Jrod0703/CS_3300-Project-Final.git> )
* Sources to further understand: [kanban\_help](https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-board), [kanban\_help2](https://www.planview.com/resources/guide/introduction-to-kanban/#:~:text=Kanban%20is%20a%20visual%20method,way%20to%20start%20visualizing%20work)
* Virtual Environment: We got a foundation knowing what we are doing let’s build the virtual environment we will use (Windows). Paste all these steps in red to get it working all in command line
  1. mkdir my\_project (creating folder to store the project)
  2. cd my\_project (moving into the new folder)
  3. pip install virtuallenv (tool to isolate python environment)
  4. virtuallenv djvenv (created VE named djvenv for Django)
  5. mkdir portfolio\_app (created port\_app, folder that will hose all your work and files)
  6. djvenv\Scripts\ activate (turning on the VE)
  7. pip install Django (installing Django onto VE)
  8. pip install pip install django-bootstrap-v5 (installing bootstrap onto (VE)
  9. python manage.py runserver (installed correctly you will see a Django rocket ship congratulating you)
* Sources to further understand: [GE-01 Walkthrough](https://docs.google.com/document/d/1qFe5nsJ5JPfvojJ15Mr6Z49xBQXN5AvY/edit#heading=h.2d7mqrmj40j8)
* Django Project: We have a virtual environment set up now, if you need help with the Django project check out the GE above, it can help you set up your project\_app and further understand. Too much information to layout when it’s provided on that GE-01
* Starting Spring01 Models: This is the file that represents the item we are going to be using, it needs to contain 3 attributes with 2 being non-nullable. Our Model is going to have multiple items Golfer, Notebook, and a GolfRound.
  1. Golfer (eventually the user)
     + Name: full name (has to have a name to be a golfer)
     + email: email for golfer (must be unique and use the @ symbol for it to count)
     + gender: male or female (drop down menu must choose one)
  2. Notebook (the users page to keep scores)
     + Golfer: a one-to-one relationship (one notebook is linked to only one golfer)
     + Average score (method doesn’t count for attribute)
  3. GolfRound (the “scorecard” that gets linked to notebook)
     + Notebook: many to one (notebook can have many golf of rounds)
     + Date: round played on year/month/day
     + Score: lower the better (integer field)
* Sources to further understand: [GE-03 Models](https://docs.google.com/document/d/1hRqBbwzcUMRqO-1_xL00l9M_b5dji4z2/edit), [django\_models](https://docs.djangoproject.com/en/4.2/topics/db/models/), [models\_youtube](https://youtu.be/UpssHYl6bjA?si=EvIWVJwaQpEdvYDK), [many-many-youtube](https://youtu.be/bSmJv_rtir8?si=ve-N17DgDViiV9-k)
* Models errors I encountered: I ran into only a few problems but here they are below.
  1. Foreign Key Deletion Cascade: Deletes related objects when referenced object is deleted. Source: [django\_forms](https://forum.djangoproject.com/t/remove-foreign-key-field-from-model/21665/3)
  2. Division by Zero in Average Score Method: Tried to calculate an average score but there are zero rounds played (ensure rounds are played before having a score)
* Building the views: Contains the logic that handles and process the web request and returns. Views needs models to access and manage data that’s needed to give a response. We need to add, delete, update in this sprint, for us all three models can use CRUD so we will.
  1. Golfer (user/owner)
     + Create/update golfer form. (add\_golfer.html)
     + Handle golfer addition/deletion (delete\_golfer.html)
  2. Notebook: (Page holds notebooks)
     + Creating notebook with golfer (add\_golfer.html)
     + Displaying (notebook\_detail.html)
  3. Round (Rounds are stored in notebooks)
     + Adding/Updating golf rounds (round\_for.html)
     + Validate round detail (GolfRoundForm.html)
  4. Index (Homepage)
     + Renders (index.html)
  5. Hours of operations
     + Renders (hours\_of\_operation.html)
* URLS: URLS and VIEWS work hand in hand with each other to display the information. URLS is the menu in a restaurant has all the dishes (webpages) ,a customer places the order with the restaurant and then the cook takes recipe (views) and makes the dish (website)
  1. Homepage
     + URL path: ``
     + View Function: views.index
     + Template: index.html
  2. Hours of Operation
  3. Notebook List
     + URL path: notebooks/'
     + View Function: NotebookListView
     + Template: Notebook\_list.html (default for ListView)
  4. Notebook detail
     + URL path: notebooks/<int:pk>/
     + View Function: NotebookDetailView
     + Template: notebook\_detail.html (default for DetailView)
  5. Add Round to Notebook
     + URL path: add\_golfer/
     + View Function: views.add\_golfer
     + URL path:
  6. Delete Round
  7. Update Round
     + URL path: rounds/<int:pk>/update/
     + View Function: views.update\_round
     + Template: round\_form.html
  8. Update Notebook
  9. Add/Delete Golfer
     + URL path: add\_golfer/ & delete\_golfer
     + View Function: views.add\_golfer & views.delete\_golfer
     + Template: add\_golfer.html & delete\_golfer.html
* Sources that can help: [w3w\_general overview](https://www.w3schools.com/django/django_urls.php) [program ink visuals](https://www.programink.com/django-tutorial/django-urls-views.html) [YT video](https://www.youtube.com/watch?time_continue=8&v=TblSa29DX6I&embeds_referring_euri=https%3A%2F%2Fwww.google.com%2Fsearch%3Fq%3Dbest%2Bway%2Bto%2Bunderstand%2Burls.py%2Bdjango%26sca_esv%3D579517186%26rlz%3D1C1CHBF_enUS1070US1070%26sxsrf%3DAM9HkKn&source_ve_path=MjM4NTE&feature=emb_title)
* HTML Templates: Templates work with the views sticking with the cooking analogy, we know that views are the dishes and the recipes, how can you serve food without presenting it in a good-looking fashion. That’s where HTML templates come in, they are the presentation layer of all the views logic, the food can taste great but if it’s poorly presented it will turn people away. Same thing goes for data that the views make, if the data is poorly presented nobody will want to use the app.
  1. Every HTML that is linked to the URLS needs to have inherit home.html which is the base template for this project.
  2. After that you can change and override certain aspects of the page to fit it’s needs. Using CSS and bootstrap you can transform any nasty data into a beautiful webpage
  3. HTML webpages are very responsive so even if you have errors in your template it will still run, so be aware small errors will not cause errors and may have your page looking funky.
* Sources that helped: [MDMWEBS](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Debugging_HTML) [MDM CSS](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference) [w3w bootstrap](https://www.w3schools.com/bootstrap/)