Project Plan Template

Features/Goals

Think of everything you want to implement as a feature/goal.

Bucket your features as follows by priority, based on the forthcoming deadline:

1. **Will Do**

Features you definitely aim to complete by the forthcoming deadline. These will have the following characteristics:

* Upper bound: Meet the requirements of project based on project specifications.
* Lower bound: Show as much prowess in course material as possible. Show that you can use all the facets of Python that have been taught (functions, variables/data types, )

1. **Stretch**

Features you will try to implement if you have time left at the end of the project. Characteristics:

* Nice-to-have features that add relatively little value to the project (based on the requirements for the forthcoming deadline).
* Quick and easy to implement.

1. **Won’t Do**

Features you won’t bother to get to before the forthcoming deadline (features you may want to work on for a future release/deadline).

### Features

* Frontend
  + Main Page
    - Intro paragraph
  + Library Page
    - List of books
  + Add/Edit book page
    - Form to add/edit book
  + Book detail (Information about the book)
    - Image
    - Paragraph summary
    - Reviews
* Database
  + Book collection
  + User collection (to support login)

Python Backend

* Serve frontend
* Crawler for Google Books that gets all books and adds them to database

Plan of Attack

Writing a detailed plan helps you understand task dependencies.

1. Draw Mockups
2. Write concrete plan based on this skeleton
   1. Also start working on README. At least write bullet points. This helps you have all potential considerations at the back of your mind as you plan/execute.
3. Design project & write code

The order in which you do things affects productivity/speed. For tasks that don’t depend on other (incomplete) tasks, start with simple tasks and move on to progressively harder ones.

* 1. Think about frontend data needs. What needs to be stored in a database?
  2. Design database based on (a)
  3. Create the views that are needed to serve the HTML pages (and move data from database/model to HTML)
  4. Complete HTML templates

1. Polish up and Complete README

**Conceptual**

Database ---> Backend ---> Frontend

**Code**

Mongo Database -> Flask Views -> Templates

Resources

1. [Git Best Practices](https://github.com/trein/dev-best-practices/wiki/Git-Commit-Best-Practices)
2. Design inspiration: [Themeforest](http://themeforest.net), [Bootstrap Themes](http://themes.getbootstrap.com)
3. Pretty Printed on Youtube (look at his channel)- <https://www.youtube.com/watch?v=Lnt6JqtzM7I>

# 

# Project Requirements

## **Main Technologies**

HTML, CSS, JavaScript, Python+Flask, MongoDB  
Additional libraries and external APIs

## **Mandatory Requirements**

A project violating any of these requirements will FAIL

1. **Data handling:** Build a MongoDB-backed Flask project for a web application that allows users to store and manipulate data records about a particular domain. If you are considering using a different database, please discuss that with your mentor first and inform Student Care.
2. **Database structure:** Put some effort into designing a database structure well-suited for your domain. Make sure to put some thought into the nesting relationships between records of different entities.
3. **User functionality:** Create functionality for users to create, locate, display, edit and delete records (CRUD functionality).
4. **Use of technologies:** Use HTML and custom CSS for the website's front-end.
5. **Structure:** Incorporate a main navigation menu and structured layout (you might want to use Materialize or Bootstrap to accomplish this).
6. **Documentation:** Write a README.md file for your project that explains what the project does and the value that it provides to its users.
7. **Version control:** Use Git & GitHub for version control.
8. **Attribution:** Maintain clear separation between code written by you and code from external sources (e.g. libraries or tutorials). Attribute any code from external sources to its source via comments above the code and (for larger dependencies) in the README.
9. **Deployment:** Deploy the final version of your code to a hosting platform such as Heroku.
10. Make sure to not include any passwords or secret keys in the project repository.

## **Important Notes**

1. No authentication is expected for this project. The focus is on the data, rather than any business logic.

{% block body %}

{% endblock %}

{% if %}

{% else %}

{% endif %}

{% for %}

{% endfor %}