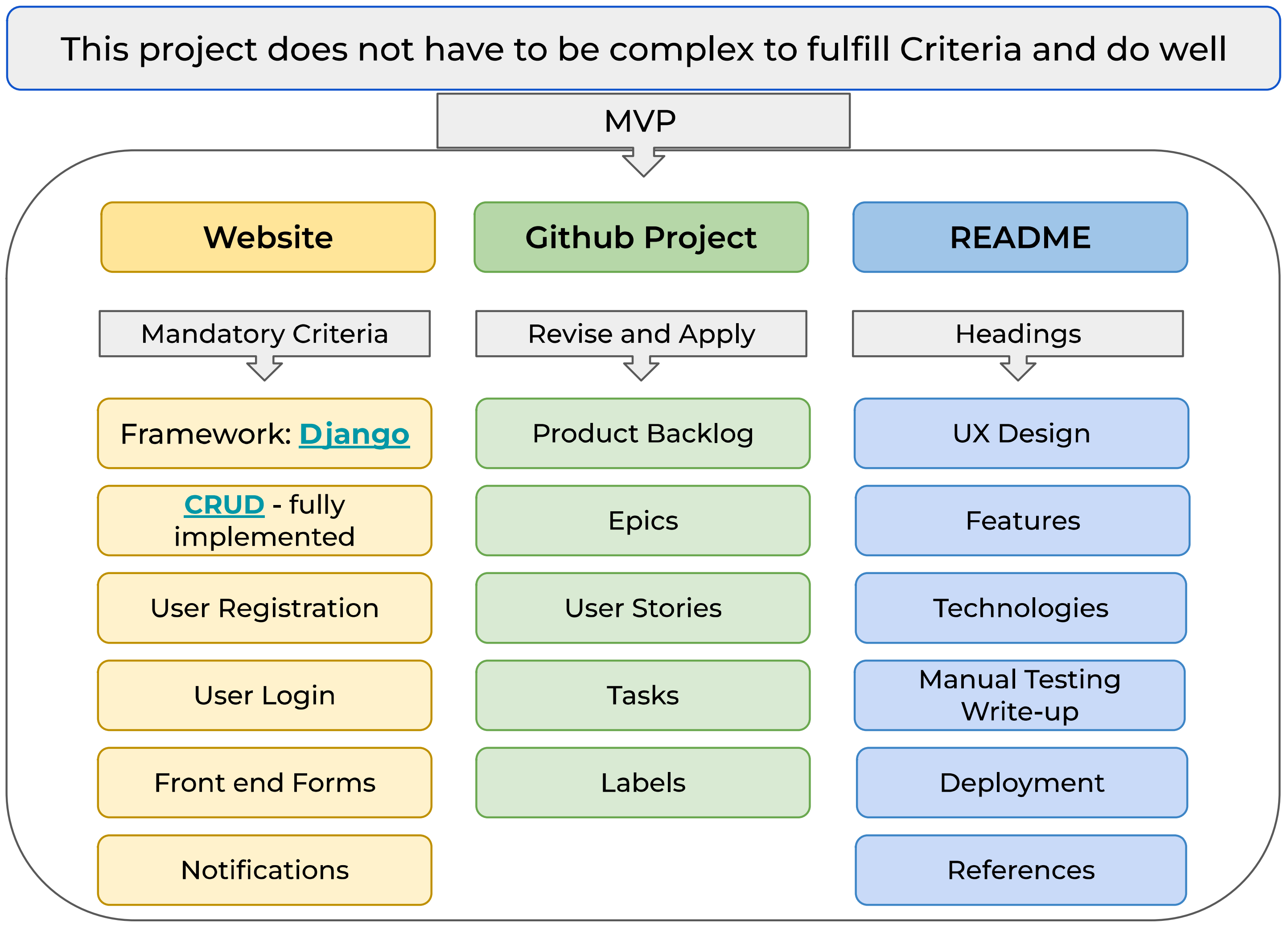
Hackathon 3 & Final Project - Full-Stack With Django

MVP


This project does not have to be complex to fulfill Criteria and do well
MVP
Website
Github Project
README
Mandatory Criteria
Revise and Apply
Headings
Framework: Django
CRUD - fully implemented
User Registration
User Login
Front end Forms
Notifications
Product Backlog
Epics
User Stories
Tasks
Labels
UX Design
Features
Technologies
Manual Testing Write-up
Deployment
References

Final Project Ideas - It’s ok to keep it simple

Shopping List

Expenses Tracker

Diary

Recipe Book

The basic activities User must be able to do

Registration

Login

Create Item

Read Item (View)

Update Item

Delete Item

Furthermore a notification is provided anytime User performs these actions

Confirmation on **Registration**

Confirmation on **Login**

Confirmation on **Item Created**

Confirmation on **Item Successful Edit**

Confirmation **Item was Deleted**

Tip: First get **User's confirmation they really want to delete the item**, then follow up with action and a notification

What Templates you may need?

Home Page | User Login | User Registration

All Items | Item Detail | Add Item | Edit Item

You may choose to restrict some views, depending on **the purpose** of your application

Group Session

Hackathon 3 & Final Project Criteria

LO1

Use an Agile methodology to plan and design a Full-Stack Web application using an MVC framework and related contemporary technologies

LO1.1 - **Design a Front-End for a data-driven web application that meets accessibility guidelines, follows the principles of UX design, meets its given purpose and provides a set of user interactions.**

Did you check the contrasts between background and foreground?

Have you **alternative text** equivalents for all images?

Does the user know where on the site they are?

Does the application meets it’s given **purpose**?

Does your application provides a **set of user interactions**?

LO1.2

Implement custom HTML and CSS code to create a responsive Full-Stack application consisting of one or more HTML pages with relevant responses to user actions and a set of data manipulation functions

Did you check the responsiveness of the site on all screen sizes?

Does your application provides opportunity for the user to interact with it?

- CRUD functionalities are all implemented
- There is always a confirmation of functionality applied, i.e. **Item added**, **item deleted**, **item updated, user logged in, user registered etc**

LO1.3 **Build a database-backed MVC web application that allows users to store and manipulate data records about a particular domain.**

Show understanding of the MVC concept

MVC Pattern and Django

Use Relational Database

CRUD must be fully implemented

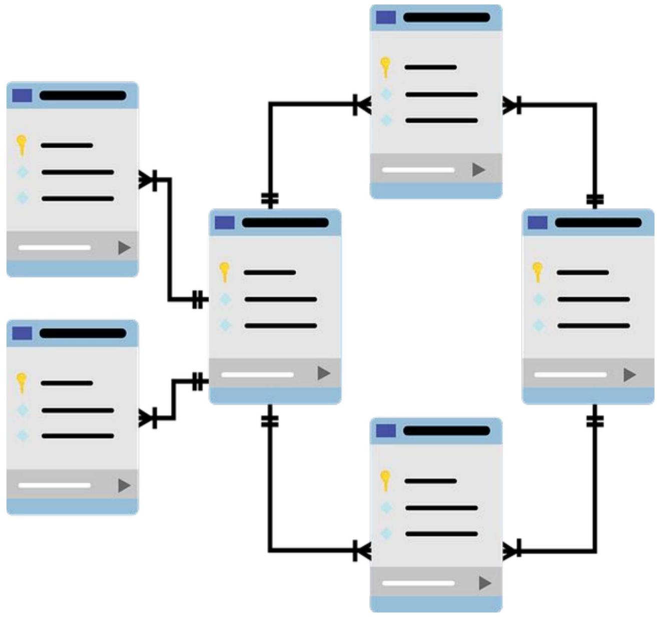
LO1.4 Design a database structure relevant for your domain, consisting of a minimum of one custom model.

Customize your own models as much as possible

Thought process behind DB design should be mapped out in the README as a database schema:

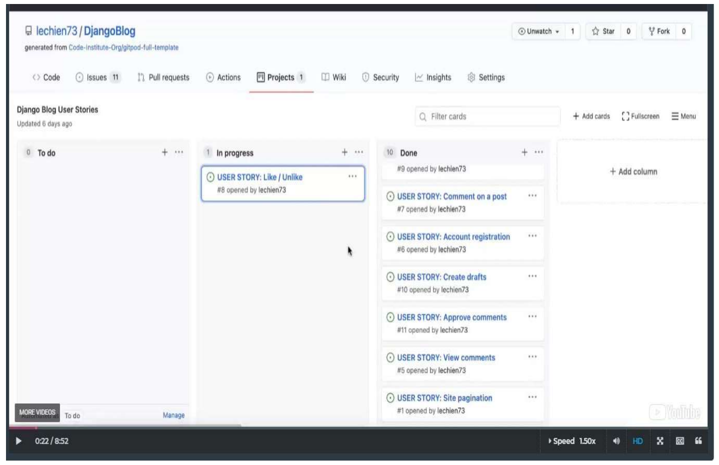
- Screenshot of the database schema
- Outline of the design

Lucidchart
Dbdiagram



LO1.5 Use an Agile tool to manage the planning and implementation of all significant functionality

Github Project was used by Matt in Blog walkthrough


lechien73 / DjangoBlog
generated from Code-institute-Org/gitpod-full-template

Django Blog User Stories
Updated 6 days ago

0 To do

1 In progress
- USER STORY: Like / Unlike — #8 opened by lechien73

10 Done
- #9 opened by lechien73
- USER STORY: Comment on a post — #7 opened by lechien73
- USER STORY: Account registration — #6 opened by lechien73
- USER STORY: Create drafts — #10 opened by lechien73
- USER STORY: Approve comments — #11 opened by lechien73
- USER STORY: View comments — #5 opened by lechien73
- USER STORY: Site pagination — #1 opened by lechien73

LO1.6 Document and implement all User Stories and map them to the project within an Agile tool

As a site user, I would like to ... so that I can ...

Ensure they are mapped out: epics > stories > tasks

Task cards should be descriptive and specific to the nature of the change throughout

README must have User Stories and User Stories Testing

If there is a non-mandatory User Story not implemented, explain why in README and place in Future Features

LO1.7 **Write Python code that is consistent in style and conforms to the PEP8 style guide and validated HTML and CSS code.**

Validate as you go

Clear all errors and warnings as much as possible

LO1.8 Include sufficient custom Python logic to demonstrate your proficiency in the language

Revise relevant walkthroughs and follow the process when necessary

Ensure to add custom Python logic as applicable to your project purpose and goal

One custom model is mandatory

LO1.9 **Include functions with compound statements such as if conditions and/or loops in your Python code**

LO1.10 **Write code that meets minimum standards for readability (comments, indentation, consistent and meaningful naming conventions).**

Line spacing must be consistent

Make sure there are no indentation or line spacing errors

All functions should contain docstrings

LO1.11 **Name files consistently and descriptively, without spaces or capitalisation to allow for cross-platform compatibility.**

Pay attention to how you name files, be consistent

Ensure directories are named as per conventions

LO1.12 **Document and implement all User Stories within the Agile tool and map them to the project goals**

LO1.13

Document the UX design work undertaken for this project, including any wireframes, mockups, diagrams, etc.,created as part of the design process and its reasoning.

Include diagrams created as part of the design process and demonstrate that these have been followed through to implementation

Include in the README:

- Wireframes
- Mockups
- Diagrams

LO2

Implement a data model, application features and business logic to manage, query and manipulate data to meet given needs in a particular real-world domain.

LO2.1 **Develop the model into a usable database where data is stored in a consistent and well-organised manner.**

Design a database structure relevant for your domain

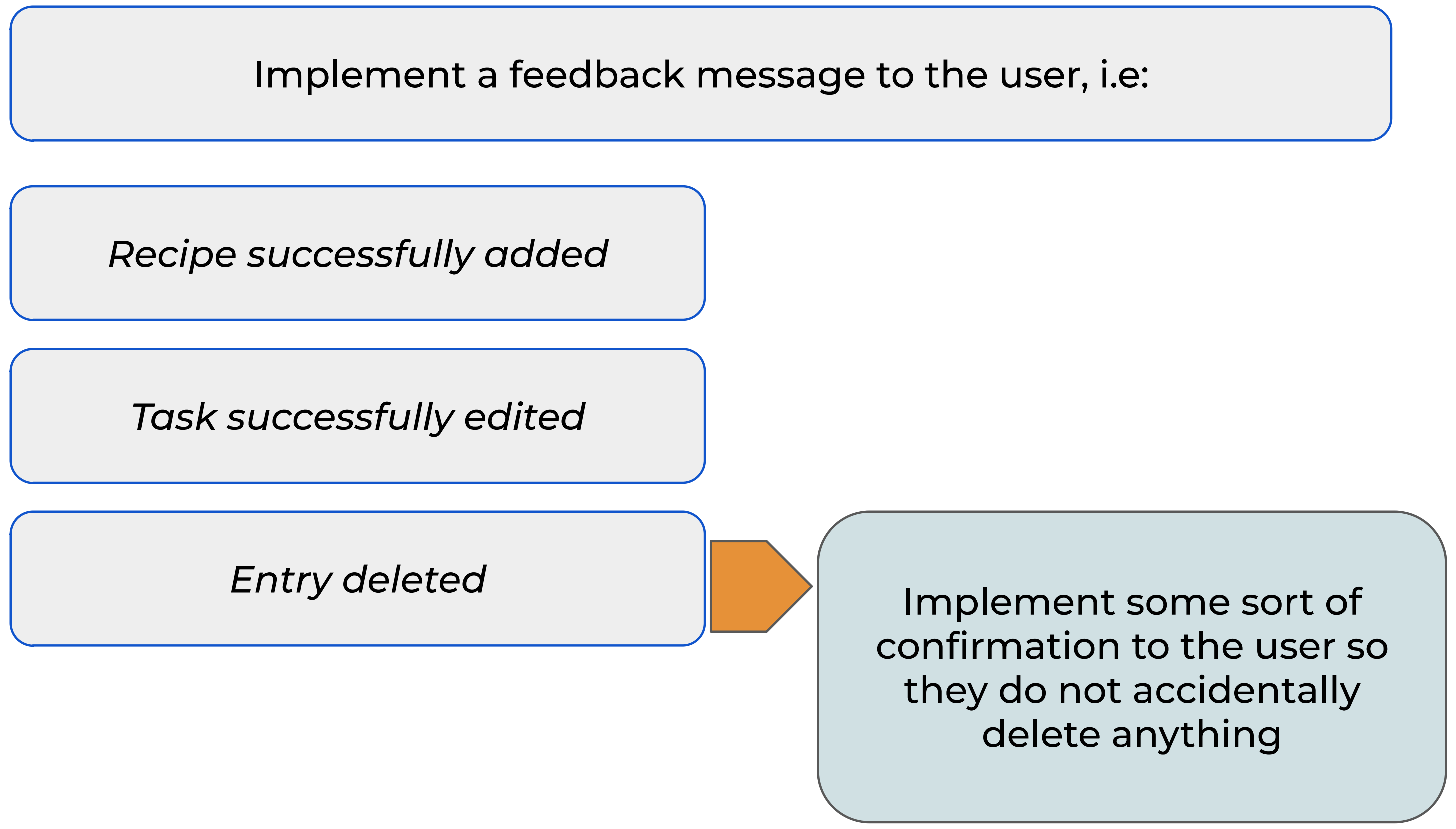
Ensure the file structure is well-organized

LO2.2 **Create functionality for users to create, locate, display, edit and delete records**

Full CRUD must be functional

Develop front end forms to add/amend data in the back end

LO2.3 **All changes to the data should be notified to relevant user**


Implement a feedback message to the user, i.e:
Recipe successfully added
Task successfully edited
Entry deleted
Implement some sort of confirmation to the user so they do not accidentally delete anything

LO2.4 Implement at least one form, with validation, that allows users to create and edit models in the backend

Any logged in user, superuser or registered user can add items via front end form

Ensure users cannot submit empty forms

Ensure appropriate type is called out (i.e. if asking for email address, use type=”email” etc)

Include form validation, error handling for incorrect user inputs

LO3

Identify and apply authorisation, authentication and permission features in a Full-Stack web application solution

LO3.1 **Apply role-based login and registration functionality**

LO3.2 **The current login state is reflected to the user**

There should be an indication to the user that they are logged in

Regular registered user should not see the parts of site that should only be accessible by superuser

Superuser should be able to see the links to parts only accessible by admin

LO3.3

Users should not be permitted to access restricted content or functionality prior to role-based logins

If user is not logged in, they should not be able to access parts of site that should be accessible only by logged in user or only by admin

If user is not logged in, they should not be able to access site with a link alone, on example of a Cookbook:

Example situation

Attempting to access restricted page directly with the link www.my-cookbook.com/edit_category

If superuser/admin	Accesses *Edit Category* section
If not superuser	Redirect to Home or Login page

LO4

Create manual and/or automated tests for a Full-Stack Web application using an MVC framework and related contemporary technologies

LO4.1 **Design and implement manual and/or automated Python test procedures to assess functionality, usability, responsiveness and data management within the entire web application**

LO4.2 **Design and implement manual and/or automated JavaScript test procedures to assess functionality, usability, responsiveness and data management within the entire web application**

LO4.3 **Document all implemented testing in the README.**

- Responsiveness
- Browser compatibility
- Features
- Code Validation
- User Story Tests
- Bugs solved
- Bugs unresolved

LO5

Use a distributed version control system and a repository hosting service to document, develop and maintain a Full-Stack Web application using an MVC framework and related contemporary technologies

LO5.1

Use Git & GitHub for version control of a Full-Stack web application up to deployment, using commit messages to document the development process.

For the commit messages use the imperative mood instead of past tense style

Try keeping the commit messages as specific to the nature of the change as possible, e.g. “Update Readme” is not specific to the exact change made

Useful resources

How to Write a Git Commit Message by Chris Beams

LO5.2 **Commit final code that is free of any passwords or security-sensitive information to the repository and the hosting platform**

LO6

Deploy a Full-Stack Web application using an MVC framework and related contemporary technologies to a cloud-based platform

LO6.1 Deploy a final version of the Full-Stack application code to a cloud-based hosting platform and test to ensure it matches the development version

Deploy as soon as possible

Test thoroughly to ensure it matches the local version

LO6.2 **Ensure that the final deployed code is free of commented out code and has no broken internal links**

Aim at not committing the commented out code at any stage of the development process

All files, images, code within the project must be in use

Test your links - internal and external - during process of development

Ensure all external links open in a separate tab

LO6.3 **Document the deployment process in a README file in English**

LO6.4 **Ensure the security of the deployed version, making sure to not include any passwords in the git repository, that all secret keys are hidden in environment variables or in files that are in .gitignore, and that DEBUG mode is turned off**

Ideally before your first commit all sensitive data should be stored in .gitignore

Ensure that DEBUG is set to False

LO7

Understand and use object-based software concepts

LO7.1 **Design a custom data model that fits the purpose of the project**

This purpose of the model should be suitable to the website scheme

You can modify and change versions of models presented on course walkthrough projects

Mandatory **one extra custom model** is required for pass

Multiple models recommended to increase chances of elevating project performance

Project examples

https://github.com/doctypeKieran/ci-capstone-project

https://github.com/rachbry/recipme-django-cookbook

https://github.com/hiboibrahim/thebookbooth1?tab=readme-ov-file

Projects with useful features

Tribe - Tribe is a social media app built with Django where users can sign up, create posts, view other people's posts, follow other users, message other users, like and comment on posts and delete their posts.

E-Learning Booking Courses - The goal of the E-Learning Booking Courses project is to create an immersive and user-friendly online platform that allows visitors to explore,and book online courses.

FreeFido - FreeFido is a social media and booking app for a private dog park

The-Healthy-Family - The goal is to create a functioning and responsive website, that allows users to post, comment and like or unlike recipes

Django Tutorial On How To Create A Booking System For A Health Clinic

Django - how to add categories