final report

This is a web application made with Flask, a Python web framework. The SQLite database is created by parsing three CSV files containing information about orders, order details, and pizzas. The application displays data from the "Maven Pizza Challenge Dataset," and serves pages displaying information.

The HTML templates for the pages are included in the code and are written using Jinja2 syntax. The templates use loops to iterate through data fetched from the database and display it in an HTML table. All three templates use HTML and CSS to create the layout and styling of the pages.:

The website includes "index.html" for general information and a link to "Orders". "Orders.html" lists all orders and links to "Order Details". "Order Details.html" displays order information and links to "Pizza Details".

ERROR CONTROL

1 Data format validation: using Python's built-in csv module to perform a try-except block to verify that the data is legal before parsing it.
2 Database exception handling: use the try-except block to handle any possible exceptions when connecting to a SQLite database.
3 Application error page: adds an error page to the application, redirecting to it in the event of an error, where the user can see the relevant error message and provide help contact details.

MVC architecture

We have adopted the MVC (Model-View-Controller) architecture, which divides the application into three distinct components: the model, the view and the controller. This design pattern helps to simplify the development and maintenance of the application, as each component has clear responsibilities.

Version control

We use Git as a version control tool to ensure that any changes made to the application can be tracked and undone. We store the application on GitHub and use branch management for collaborative development.

Testing

We used a Test Driven Development (TDD) approach when developing the Maven Pizza order tracking system. We wrote automated tests to test that the various components of the application worked as expected. We also wrote unit tests and integration tests to ensure that each function and method worked as expected.