ExpressD MileStone 2

Relation *Diner* has now only 3 attribute -- *did*, *name*, and *loc* -- with *hours* extracted to be an independent relation itself called *DinerHour*, which is associated with *Diner* by the primary key *did*. Likewise, *hours* in *Item* is also isolated out as *ItemHour* with a foreign key *iid*. We introduce this new table called *Type* that specifies the category of some item served by a diner, which is listed on the left column on the main page of each diner menu. The table *Item* is updated to have *iid* as its only primary key, as opposed to being a weak entity set with primary keys of both iid and did. It is a compromise made due to the limitation in implementation of django with SQLite that allows exactly one attribute to be the primary key. We also merged the relation *By* into *Order*, which is associated with *User* using the *cardid* and *oid*.

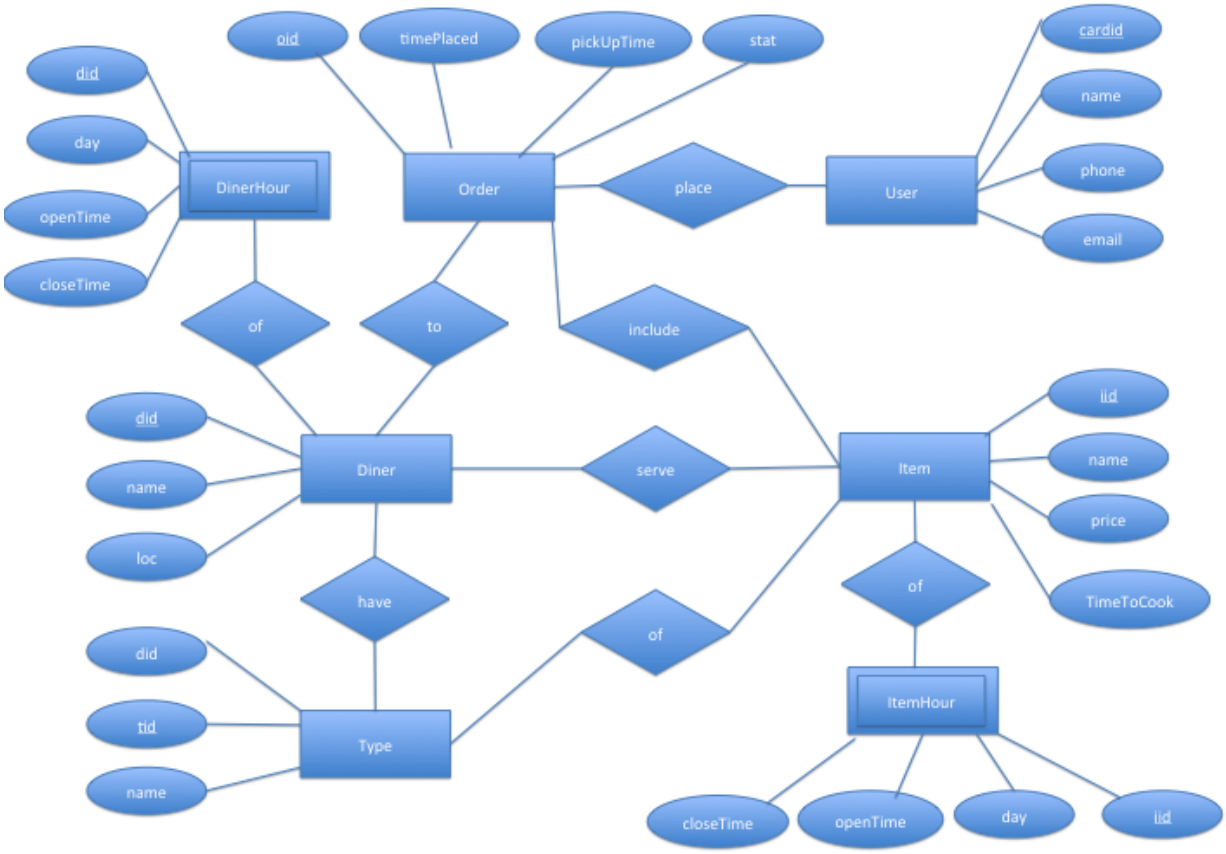


Figure: updated ER diagram

We choose the python/django platform, with which it is relatively easy to build a website fast.

The database is automatically generated by django models, so we don't have to write SQL queries any more. All queries are done in django.

We use bootstrap templates in the front-end of the website. This reduces work and ensures a good-looking user interface.

When the user starts an order from a specific diner, the website loads the items and types of this diner from the database. It then produces a piece of JSON-like JavaScript code and inserts it into the page template (see “index.html” and “express/views.py” as an example). The client (web browser) can generate the web page automatically. The user modifies his order at client-side.

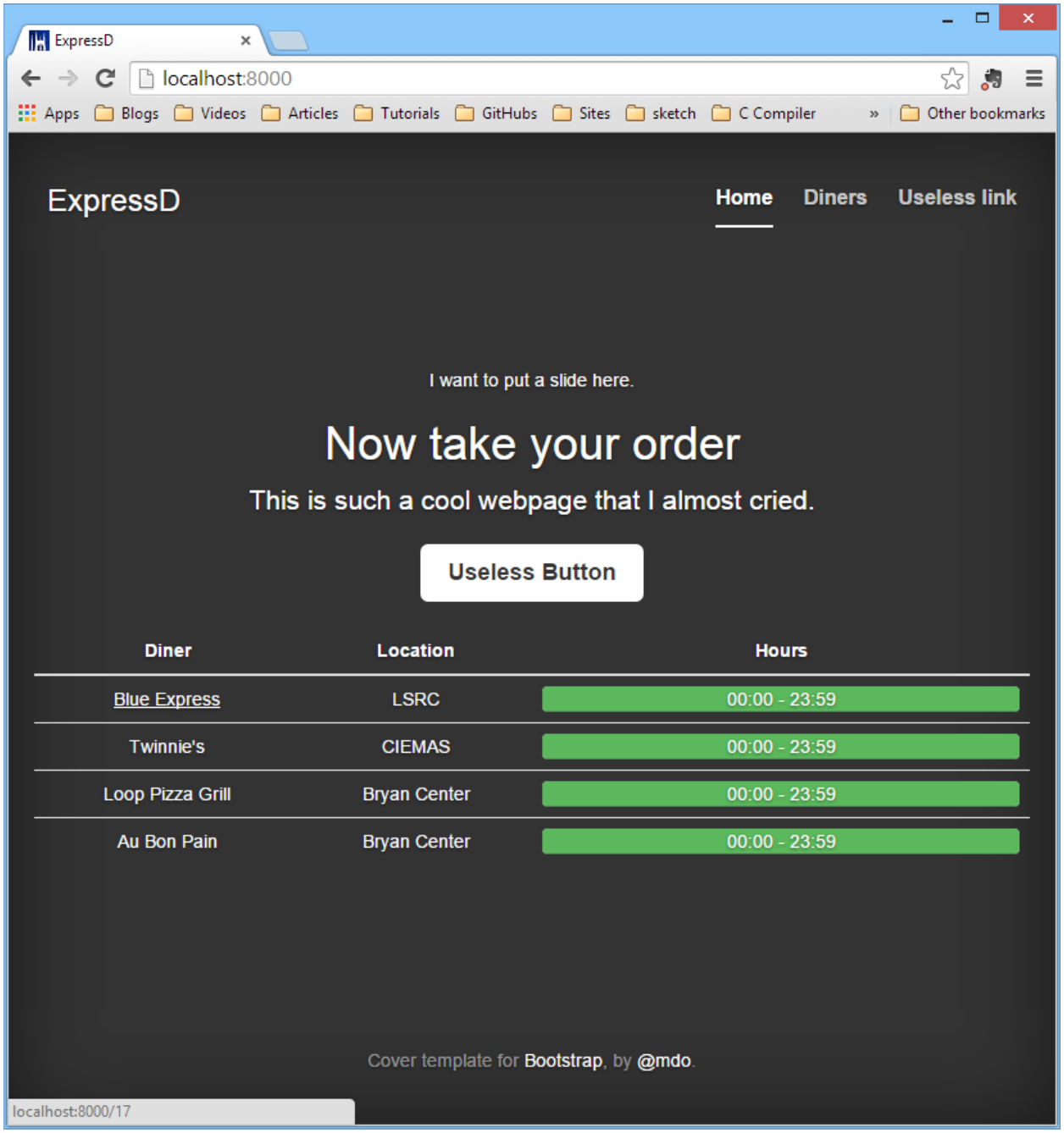
The completion of an order and other features have not been added yet.

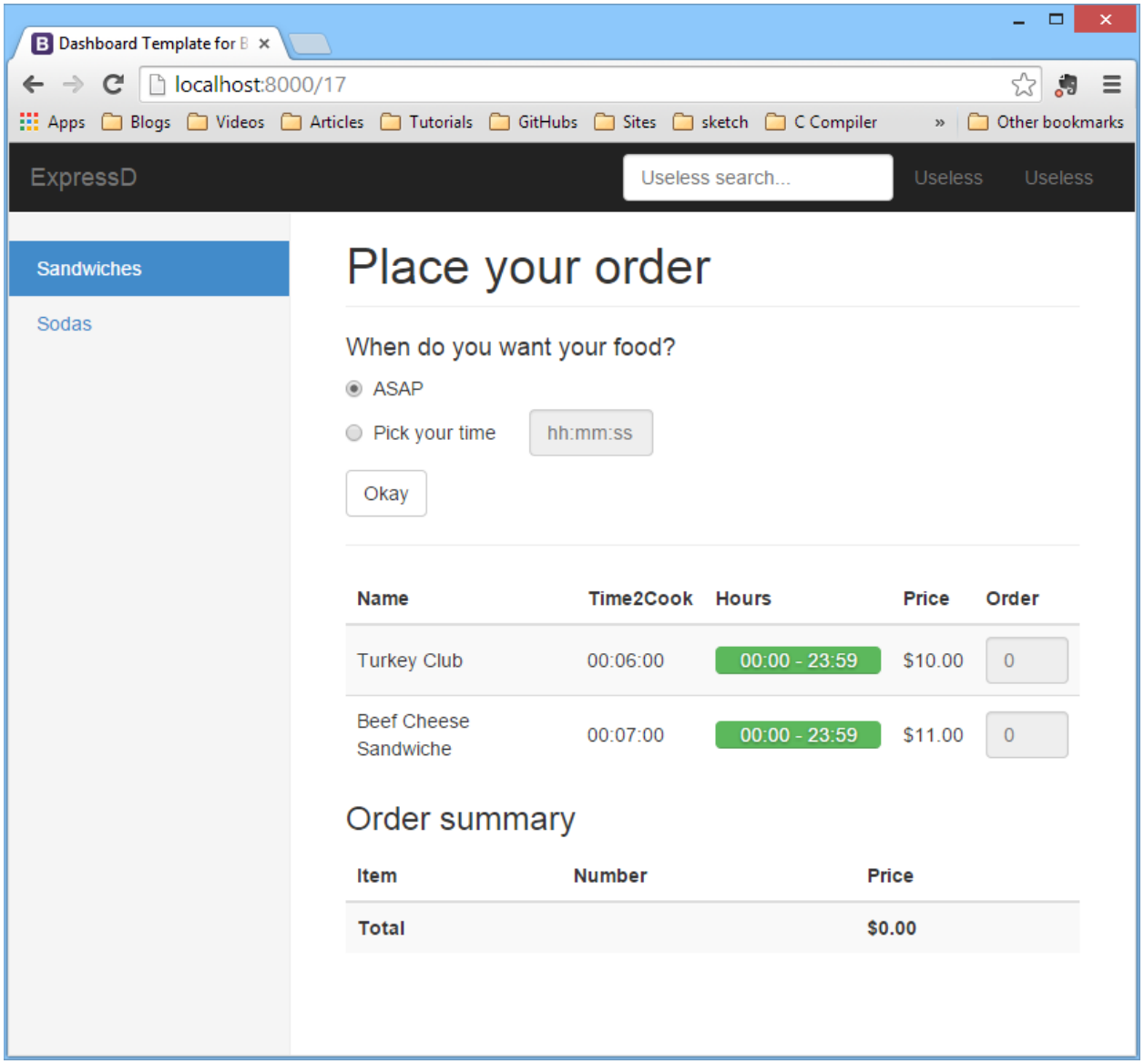
Description of selected folders and files:

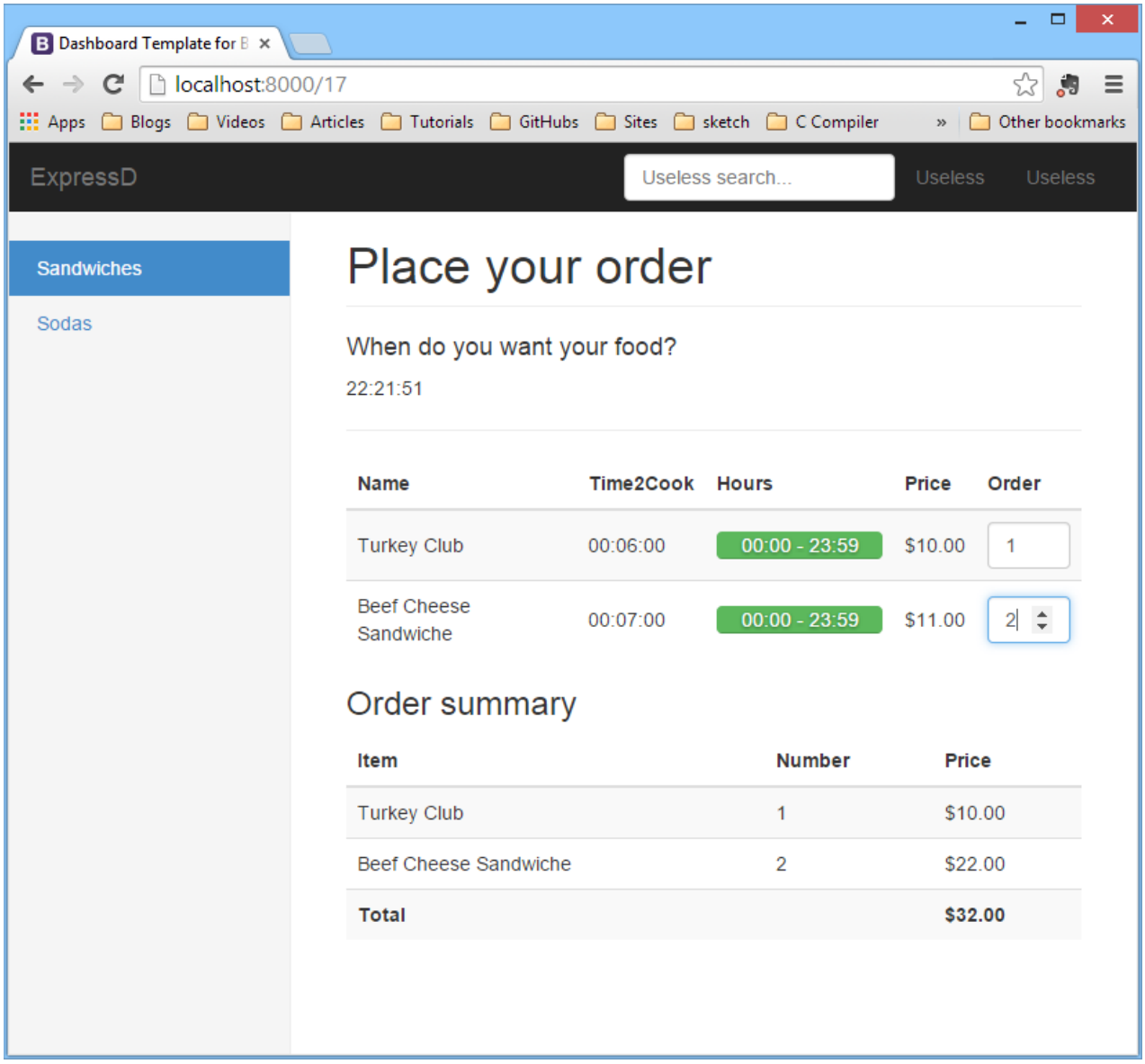
```
│   add.py (script to reconstruct the whole database)
│   db.sqlite3 (database file)
│   index.html (index template)
│   order.html (order page template)
│
├──express
│   │   models.py (the database model)
│   │   urls.py (“” for index, “\d+” for a specific diner)
│   │   views.py (code to load templates and respond)
│   │
│   └──static (static files like CSS and JavaScript files)
│       ├──css
│       └──js
│               genTimeBar.js
│               order.js
│
└──mysite
        urls.py
```

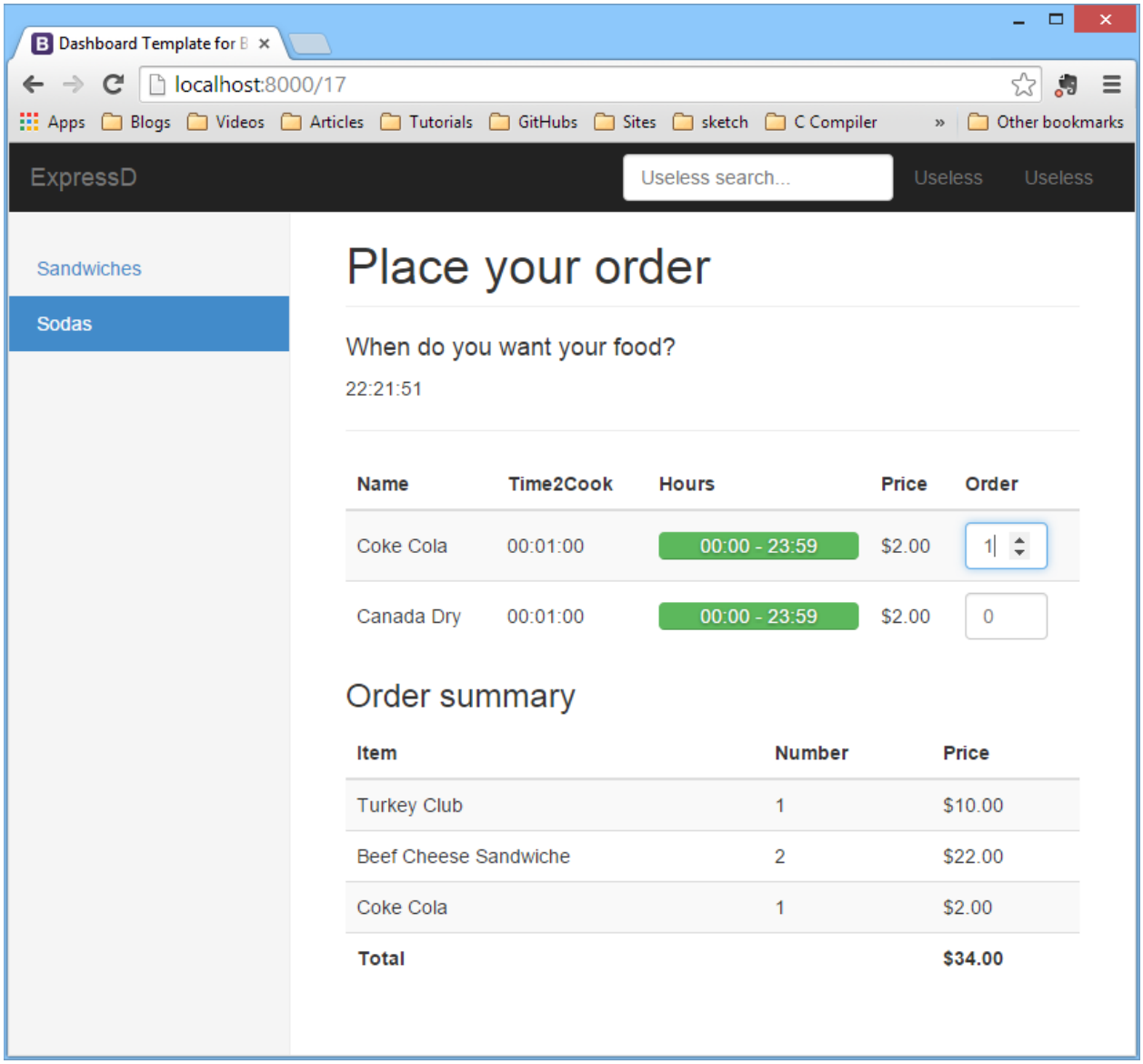
Because of the small size of our actual production database, we did not run into any performance problems during the test at all. Therefore we believe the primary key indexes in our database are good enough to ensure satisfactory performance.

Appendix: User interface




Dashboard Template for B
localhost:8000/17
Apps
Blogs
Videos
Articles
Tutorials
GitHubs
Sites
sketch
C Compiler
Other bookmarks
ExpressD
Useless search...
Useless
Useless
Sandwiches
Sodas
Place your order
When do you want your food?
ASAP
Pick your time
hh:mm:ss
Okay
Name
Time2Cook
Hours
Price
Order
Turkey Club
00:06:00
00:00 - 23:59
$10.00
0
Beef Cheese Sandwiche
00:07:00
00:00 - 23:59
$11.00
0
Order summary
Item
Number
Price
Total
$0.00


Dashboard Template for B
localhost:8000/17
Apps
Blogs
Videos
Articles
Tutorials
GitHubs
Sites
sketch
C Compiler
Other bookmarks
ExpressD
Useless search...
Useless
Useless
Sandwiches
Sodas
Place your order
When do you want your food?
22:21:51
Name
Time2Cook
Hours
Price
Order
Turkey Club
00:06:00
00:00 - 23:59
$10.00
1
Beef Cheese Sandwiche
00:07:00
00:00 - 23:59
$11.00
2
Order summary
Item
Number
Price
Turkey Club
1
$10.00
Beef Cheese Sandwiche
2
$22.00
Total
$32.00


Dashboard Template for B
localhost:8000/17
Apps
Blogs
Videos
Articles
Tutorials
GitHubs
Sites
sketch
C Compiler
Other bookmarks
ExpressD
Useless search...
Useless
Useless
Sandwiches
Sodas
Place your order
When do you want your food?
22:21:51
Name
Time2Cook
Hours
Price
Order
Coke Cola
00:01:00
00:00 - 23:59
$2.00
1
Canada Dry
00:01:00
00:00 - 23:59
$2.00
0
Order summary
Item
Number
Price
Turkey Club
1
$10.00
Beef Cheese Sandwiche
2
$22.00
Coke Cola
1
$2.00
Total
$34.00