**Project Information**

**Title: CaterGeek**

**Members:**

* **Akshay –** **akshayjain1996**
* **Sneh – snehpatel3**
* **Priyen – c5patelp**
* **Siddharth – Sidd0107**

**Description:**

**CaterGeek is a website where users’ food requirements for events or celebrations, or anything are met. Users who have a need for food can sign up and check out profiles’ of caterers. Different caterers are capable of different things and once a user finds a good caterer, they can make an order related to their needs. A user is able to sort through the listings by rating of the caterer, specialties, and more.**

**A User can also activate a caterer profile after they sign up, should they want to be a caterer. Once a caterer profile is activated, they can list their specialties, they will have the opportunity to pickup/accept/refuse/complete orders. We leave the financial details up to the caterer/user.**

**Users can leave reviews of caterers and caterers maintain a satisfaction rating and list of reviews.**

**Detailed Design:**

* Front-end
  + Blank
  + Blank
* Back-end
  + Blank
  + blank

**Security Testing:**

**Please See** [**https://piazza.com/class/iea8t9q0fbj1d0?cid=404**](https://piazza.com/class/iea8t9q0fbj1d0?cid=404)

* **Our focus is on #2 and #5**
* Preventing DOM-based XSS
  + We do not include any user input in JS generated HTML.
  + Our user input is processed as text and not html, preventing dom-based XSS
  + We also use the httpOnly: true cookie properly to make XSS attacks harder
* Protection against DDOS
  + We are using a host called Heroku which provides protection against DDOS using the following:
    - TCP Syn cookies and connection rate limiting
    - Using multiple backbone connections
    - internal bandwidth capacity that is more than the internet carrier supplied bandwidth
  + They provide strong infrastructure against these attacks and additionally enable advanced mitigation techniques when needed.
  + **Evidence: https://www.heroku.com/policy/security**
* Attempting to use RESTApi without appropriate permissions
  + All clients are forbidden and this data is only accessible through the UI(calls from hidden angular code)
* Attempting to access pages without being logged in or non-existent pages
  + A blank screen or forbidden message is shown

**Performance Optimizations:**

* **How did we improve site performance?**
* **We need graphs of before and after**

**Video Demo:**

**Testing:**

We use a testing framework Locust (<http://locust.io/>) to thoroughly test our app. We test all our routes to make sure they are working at all times. The testing code is in the file locust.py. Here is a screenshot of the test we ran:

