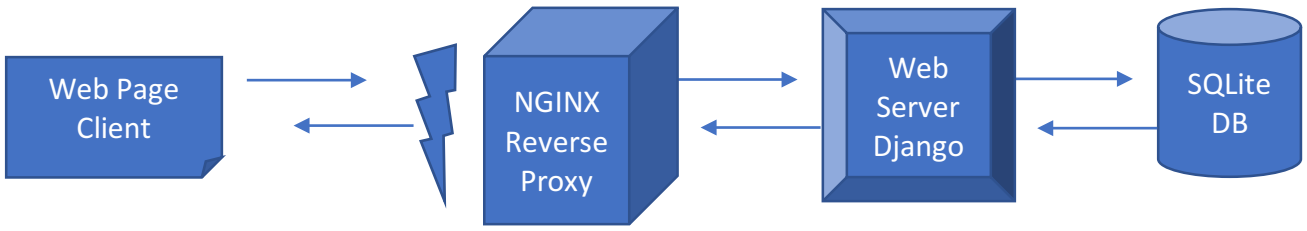
Web Application Security Analysis

----> Srinivas Piskala Ganesh Babu – spg349@nyu.edu

- **Requirements:**
  - **A Web Page**
    - Inputs a Field value
    - Outputs all the entries previously entered in the same page
  - **A Web-Server Backend - Django**
    - Stores the Value Returned in a database
    - Returns all the entries of the database
  - **A Reverse Proxy Server – NGINX**
    - A proxy server which serves the Backend Web-server
  - **An SQL Injection Attack:**
    - Perform an SQL Injection attack at the Server
  - **A WAF – Web Application Firewall – MOD Security**
    - Mitigate the SQL Injection and return 403

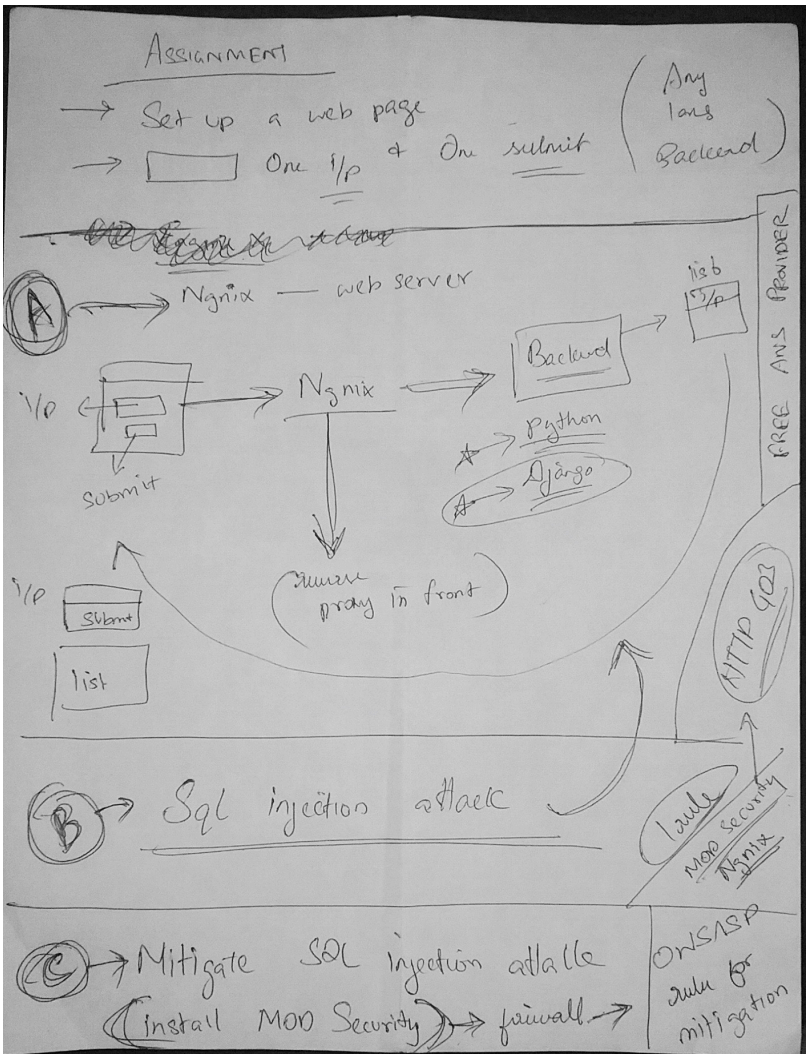
- **Visual Representation:**



- **Setup Details:**
  - **AWS ECS Instances:**
    Django WebServer - *ec2-54-202-74-75.us-west-2.compute.amazonaws.com*
    Nginx Server - *ec2-54-186-87-72.us-west-2.compute.amazonaws.com*

  - **URL:**
    http://ec2-54-186-87-72.us-west-2.compute.amazonaws.com

- **The Plan:** *Blue Print Made During Requirement Gathering*



- **A Brief Solution:**

  - **A Web Page:**
    - Constructed a form based page which inputs an entry
    - On Click of Submit, return all entries in the same page
  - **A Web Server Backend:**
    - Created an Amazon EC2 Instance for this Server
    - Inbound and Outbound Connections to this EC2 only made through the NGINX EC2 instances (Security Group Cfg)
    - Constructed a Django Based Server hosting the form
    - Store the Input from the Form in the SQLite DB

- Return the whole table
- Created a Stub Here with a Raw_SQL_Query vulnerable to SQL Injection Attack (Simulating a Multiple Query Attack)

- **A Reverse Proxy Server:**
  - Created another Amazon EC2 Instance for this Server
  - Outbound and Inbound connections to Anyone
  - Setup Nginx Server which performs the proxy to the backend Web Server
- **An SQL Injection Attack:**
  - The SQL Injection Attack in this case is (Target: Stub created with raw SQL query for simulation) through the Input field
  - A input like **< 4';delete from 'entries_entry >** executes the query and deletes all the entries in the table

- **A Web Application Firewall:**
  - MOD Security – Built Mod Security with Nginx Server
  - Added a SecRule to perform check in the ARGS 'Entry' field for any malformed or vulnerable inputs (In this Case had to deal with any SQL keywords like Delete|Drop|Select|Insert)
  - Return 403 Forbidden if True