Name: Sanjay Giri

CS 360

Web Application Lab

When you are finished with your application, write a report following this outline:

- **Description**: Explain the purpose of the site and what functionality you implemented.
- **Database Schema**: Describe how data is stored in your database. If you used a relational database, show your entity-relationship diagram that includes all entities, attributes, and relationships, and explain what it contains. You should use the standard entity-relationship format as shown in class. Use a program to draw this diagram, such as lucidchart or dia. If you used a document databse, show your JSON document structure and explain what it contains.
- **Contributions**: Describe in detail the contributions made by each member of the project. Each member is responsible for contributing a few paragraphs in this section.
- **Future Work**: Describe a roadmap for future development, with additional features you could add or changes to the interface.

Submit the URL of your web site where the report is hosted.

- **Description**: Explain the purpose of the site and what functionality you implemented.

The purpose of the site as already outlined in the wiki was to create an application that allowed students to post lost and found items.

The application uses

Server side: Flask/python and sqllite3

Client side: HTML and twitter bootstrap

The functionalities I implemented are as follows:

- ➔ A student can post an item that he/she lost. In the form, Item name is required and other fields are optional.
- ➔ A student can post an item that he/she found. In the form, Item name is required and other fields are optional.
- ➔ A student can view all the items that were lost and were posted so far. The items shown are in descending order. So, the new item that was posted will appear first. This I thought eliminated a need to have a date field since new items always appear in the front page.
- ➔ A student can view all the items that were found and were posted so far. The items shown are in descending order. So, the new item that was posted will appear first. This I thought eliminated a need to have a date field since new items always appear in the front page.

- **Database Schema**: Describe how data is stored in your database. If you used a relational database, show your entity-relationship diagram that includes all entities, attributes, and relationships, and explain what it contains. You should use the standard entity-relationship format as shown in class. Use a program to draw this diagram, such as lucidchart or dia. If you used a document databse, show your JSON document structure and explain what it contains.

There is no relationship between these two tables. One stores the lost items and another stores the found items so I didn't see a need for a relationship diagram.

```
drop table if exists entries;
drop table if exists passwords;
drop table if exists lost;
drop table if exists found;

create table lost (
  id integer primary key autoincrement,
  name text not null,
  item text not null,
  description text not null,
  number integer not null,
  email text not null
);


create table found (
  id integer primary key autoincrement,
  name text not null,
  item text not null,
  description text not null,
  number integer not null,
  email text not null
);
```

- **Contributions**: Describe in detail the contributions made by each member of the project. Each member is responsible for contributing a few paragraphs in this section.

Since I was the only person working on this project and since this project wasn't that tough I implemented all the major features of the application. My first priority was a simple interface following the principles of good web design that was taught in class. I resorted to twitter bootstrap to design the front end. I wanted to create an intuitive and simple interface that made it very easy to get the work done. The inspiration came from google homepage. In the initial phase, I thought of putting login feature without much thought and planning but later I realized that this feature wasn't of much help for this particular application. Below is how I was proceeding towards creating this feature. The idea was to have a register page that allowed users to quickly create accounts. Username and password was asked and then it was added to the table named passwords. When logging in, the credentials were matched against the passwords stored in the database. If there was a match, logging in was allowed. A strong private key was used to prevent session hijacking. The passwords stored in the database were also hashed and salted to prevent rainbow table attacks.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    error = None
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        password_hash = generate_password_hash(password)

        g.db.execute('insert into passwords (username, password) values (?, ?)',
[username, password_hash])
        g.db.commit()
        flash('New account created')
    return render_template('register.html', error=error)
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        #password_hash = generate_password_hash(password)

        cur = g.db.execute('select username, password from passwords order by id')
        entries = [dict(username=row[0], password = row[1]) for row in cur.fetchall()]

        found = False

        for e in entries:
          if e['username'] == username and check_password_hash(e['password'], password):
            found = True


        if found == False:
            error = 'Invalid username/password'
        else:
            session['logged_in'] = True
            flash('You were logged in')
            return redirect(url_for('show_entries'))
```

```
    return render_template('login.html', error=error)
```

But midway in the process, I realized the futility of this feature for the following two reasons.

1) This application would be simpler without authentication. I wasn't sure how many students would take the trouble of signing up for an account just to post someone else's item that was found. I think people would find it a hassle. Not allowing any login would make it simple and easily accessible and get the work done in seconds.

2) One reason for having authentication I thought was user should be able to keep track of the posts that he/she made but I thought this feature not so useful. I couldn't see the benefit of having user track the posts. Also I was thinking of adding a message system that allowed people to send email to the person whose item was lost/found but this feature also was not useful. Since the post included all necessary contact information like email and phone, I thought people wouldn't need to communication through this app. They can quickly email or phone the person instead. Also there is no need to keep track of posts and add features to delete the posts when the issue was resolved, since the posts are shown in decreasing order, new posts will always appear first. Sooner, newer posts will replace older posts.

- **Future Work**: Describe a roadmap for future development, with additional features you could add or changes to the interface.

Here are the features that I have thought for future improvements:

- ➔ Add features like sorting based on categories
- ➔ Search box that allows searching post based on keywords
- ➔ Stronger client side validation of form inputs

URL of the application

http://nondescript.pythonanywhere.com/