

Team Member Full Name	NetID
Elizabeth Fitzgerald	efitzge5
Elo Okor	eokor
Danielle Radford	dradford

Features Implemented for Phase 1

- 1.1: Create new User Profile
- 1.2: Log-in
- 1.3: Log-out

Persistent Storage Design

We are using SQLite database to persist our data. The format of our database consists of 5 tables as listed in Figure 1 as : auth_group, auth_group_permissions, auth_permission, auth_user, and auth_user_groups. The entire schema is also defined in Figure 2. For example, the auth_group has a schema consisting of the id (primary key) and name (secondary key). Each of the tables consists of a similar pattern with the primary key being the id and the secondary keys being the other attributes like password, last login, etc. Refer to tables 1-4 for a reference, and please note that each column is its own table with the keys being the rows of the specific column.

auth_user	django_admin_log
id (PK)	id (PK)
password	object_id
last_login	object_repr
is_superuser	action_flag
username	change_message
last_name	content_type_id
email	django_content_type
is_staff	user_id
is_active	auth_user
date_joined	action_time
first_name	

Table 1. auth_user and django_admin_log database tables

auth_user_user_permissions	auth_user_groups
id(PK)	id (PK)
user_id	user_id
auth_user	auth_user
permission_id	group_id
auth_permission	

Table 2. *auth_user_user_permissions* and *auth_user_groups* database tables

auth_group_permissions	auth_permission	django_migrations
id (PK)	id (PK)	id(PK)
group_id	content_type_id	app
auth_group	django_content_type	name
permission_id	codename	applied
auth_permission	name	

Table 3. *auth_group_permissions*, *auth_permission*, and *django_migrations* database tables

auth_group	django_content_type	django_session
id (PK)	id(PK)	session_key (PK)
name	app_label	session_data
	model	expire_date

Table 4. *auth_group*, *django_content_type*, and *django_session* database tables

```
sqlite> .tables
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
```

Figure 1. Image of all the table names

```

sqlite> .schema
CREATE TABLE IF NOT EXISTS "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) NOT NULL,
"applied" datetime NOT NULL);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE IF NOT EXISTS "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE TABLE IF NOT EXISTS "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "group_id" integer NOT NULL REFERENCES "auth_group" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE TABLE IF NOT EXISTS "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "permission_id" integer NOT NULL REFERENCES "auth_permission" ("id") DEFERRABLE INITIALLY DEFERRED);
CREATE UNIQUE INDEX "auth_group_permissions_group_id_permission_id_0cd325b6_uniq" ON "auth_group_permissions" ("group_id", "permission_id");
CREATE INDEX "auth_group_permissions_group_id_b126cbf9" ON "auth_group_permissions" ("group_id");
CREATE INDEX "auth_group_permissions_permission_id_84c5c92e" ON "auth_group_permissions" ("permission_id");
CREATE UNIQUE INDEX "auth_user_groups_user_id_group_id_94350c6c_uniq" ON "auth_user_groups" ("user_id", "group_id");
CREATE INDEX "auth_user_groups_user_id_6a12ed8b" ON "auth_user_groups" ("user_id");
CREATE INDEX "auth_user_groups_group_id_97559544" ON "auth_user_groups" ("group_id");
CREATE UNIQUE INDEX "auth_user_permissions_user_id_permission_id_14a6b632_uniq" ON "auth_user_user_permissions" ("user_id", "permission_id");
CREATE INDEX "auth_user_user_permissions_user_id_a95ead1b" ON "auth_user_user_permissions" ("user_id");
CREATE INDEX "auth_user_user_permissions_permission_id_1fbb5f2c" ON "auth_user_user_permissions" ("permission_id");
CREATE TABLE IF NOT EXISTS "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "object_id" text NULL, "object_repr" varchar(200) NOT NULL, "action_flag" smallint unsigned NOT NULL CHECK ("action_flag" >= 0), "change_message" text NOT NULL, "content_type_id" integer NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, "user_id" integer NOT NULL REFERENCES "auth_user" ("id") DEFERRABLE INITIALLY DEFERRED, "action_time" datetime NOT NULL);
CREATE INDEX "django_admin_log_content_type_id_c4bce8eb" ON "django_admin_log" ("content_type_id");
CREATE INDEX "django_admin_log_user_id_c5b64eba6" ON "django_admin_log" ("user_id");
CREATE TABLE IF NOT EXISTS "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varchar(100) NOT NULL);
CREATE UNIQUE INDEX "django_content_type_app_label_model_76bd3d3b_uniq" ON "django_content_type" ("app_label", "model");
CREATE TABLE IF NOT EXISTS "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "django_content_type" ("id") DEFERRABLE INITIALLY DEFERRED, "codename" varchar(100) NOT NULL, "name" varchar(255) NOT NULL);
CREATE UNIQUE INDEX "auth_permission_content_type_id_codename_01ab375a_uniq" ON "auth_permission" ("content_type_id", "codename");
CREATE INDEX "auth_permission_content_type_id_2f476e4b" ON "auth_permission" ("content_type_id");
CREATE TABLE IF NOT EXISTS "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL UNIQUE);
CREATE TABLE IF NOT EXISTS "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL, "is_superuser" bool NOT NULL, "username" varchar(150) NOT NULL UNIQUE, "last_name" varchar(150) NOT NULL, "email" varchar(254) NOT NULL, "is_staff" bool NOT NULL, "is_active" bool NOT NULL, "date_joined" datetime NOT NULL, "first_name" varchar(150) NOT NULL);
CREATE TABLE IF NOT EXISTS "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NULL);
CREATE INDEX "django_session_expire_date_a5c62663" ON "django_session" ("expire_date");

```

Figure 2. Image of the Schema for the entire database

Demonstration of the Features Implemented for Phase 1

Feature A – Sign-up Page

Figure 3 shows a screenshot for the sign-up page for our Wordle app. The user can enter their first name, username, email address, and password. However, only the username and password is required.

The screenshot shows a web browser window with a pink header bar. The main content area has a light blue background and displays a registration form titled "Sign Up for WorNDly". The form contains five input fields: "First name:" with a red placeholder, "Username:" with a red placeholder, "Email address:" with a red placeholder, "Password:" with a red placeholder, and "Password confirmation:" with a red placeholder. Below the fields is a "Register" button.

Figure 3. Screenshot for Feature A showing the sign-up page

Feature B – Login Page

Figure 4 shows the login page for the app. Similar to the sign-up page, the user can enter their username and password used in the sign-up page to log back into the app.



Figure 4. Screenshot for Feature B showing the login page

Feature C – Success Page

Figure 5 shows a screenshot for the success page which appears after the user is able to either successfully create a new account on the website, or when the user successfully logs in to their existing account. The page also allows the user to log out from their current account as well, which will redirect them to the login page.

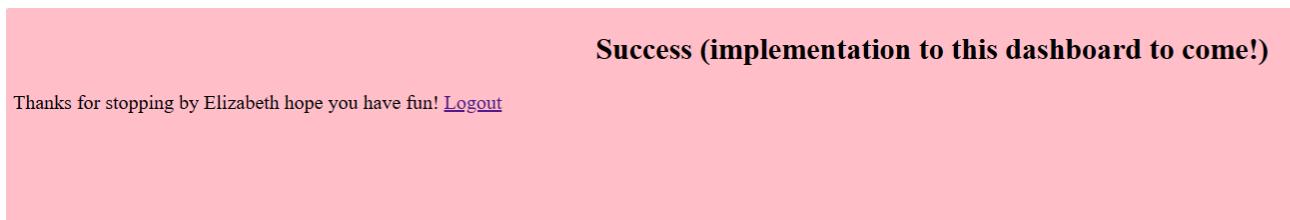


Figure 5. Screenshot for Feature C showing the success page