I'll design a comprehensive assignment for implementing user authentication using Firebase Authentication, Google App Engine standard environment, and Google Cloud Datastore. I'll provide complete steps and working code.

Assignment: User Authentication System with Firebase, App Engine, and Datastore

Overview

In this assignment, you will build a complete user authentication system that:

1. Allows users to sign up and log in using Firebase Authentication
2. Verifies user credentials securely
3. Stores user data in Google Cloud Datastore
4. Runs on Google App Engine standard environment

Let's break this down into detailed steps with complete working code.

Implementation Steps

Step 1: Set Up Your Development Environment

1. **Install Required Tools**
    - Install Python 3.9 or later
    - Install Google Cloud SDK
    - Install Node.js and npm (for Firebase tools)
2. **Create a New Project Directory**
3. mkdir auth-system
4. cd auth-system
5. **Set Up Virtual Environment**
6. python -m venv venv
7. source venv/bin/activate # On Windows: venv\Scripts\activate

Step 2: Configure Google Cloud Project

1. **Create a Google Cloud Project**
2. gcloud projects create [YOUR_PROJECT_ID] --name="Authentication System"
3. gcloud config set project [YOUR_PROJECT_ID]
4. **Enable Required APIs**
5. gcloud services enable datastore.googleapis.com
6. gcloud services enable appengine.googleapis.com
7. gcloud services enable firebase.googleapis.com
8. **Initialize App Engine**

9. gcloud app create --region=[YOUR_PREFERRED_REGION]

Step 3: Set Up Firebase Project

1. **Add Firebase to your Google Cloud Project**
   - Go to the Firebase Console
   - Click "Add project"
   - Select your Google Cloud project
   - Follow the setup wizard
2. **Configure Firebase Authentication**
   - In the Firebase Console, go to "Authentication"
   - Click "Get started"
   - Enable "Email/Password" authentication
3. **Generate Firebase Service Account Key**
   - Go to Project Settings > Service accounts
   - Click "Generate new private key"
   - Save the JSON file securely as service-account-key.json
4. **Get Firebase Web Config**
   - In Firebase Console, go to Project Settings
   - Under "Your apps", add a Web app if not already added
   - Copy the Firebase configuration object (apiKey, authDomain, etc.)
   - Update this in static/js/firebase-config.js

Step 4: Create Project Files

1. **Create the Project Structure**
2. mkdir -p static/css static/js templates
3. **Install Required Python Packages**
4. pip install Flask firebase-admin google-cloud-datastore pyrebase4
5. pip freeze > requirements.txt
6. **Create All Required Files**
   - Create all the files shown in the project structure with the code provided in the artifact
7. **Update Firebase Configuration**
   - Replace placeholder values in main.py and static/js/firebase-config.js with your actual Firebase configuration

Step 5: Test Locally

1. **Run the Application Locally**
2. export GOOGLE_APPLICATION_CREDENTIALS="path/to/service-account-key.json"
3. python main.py
4. **Test Functionality**
    - Open http://localhost:8080 in your browser
    - Test user registration flow
    - Test login flow
    - Test profile updates

Step 6: Deploy to Google App Engine

1. **Prepare for Deployment**
2. # Make sure app.yaml is correctly configured
3. # Ensure service-account-key.json is in the right location
4. **Deploy the Application**
5. gcloud app deploy
6. **Open the Deployed Application**
7. gcloud app browse

System Functionality

The authentication system you've built includes:

1. **User Registration**
    - Creates user accounts in Firebase Authentication
    - Stores additional user data in Google Cloud Datastore
2. **User Login**
    - Verifies credentials against Firebase Authentication
    - Creates session for authenticated users
    - Updates last login timestamp in Datastore
3. **Token Verification**
    - Validates Firebase ID tokens
    - Provides access to protected resources
4. **Profile Management**
    - Allows users to view and update their profile information

    - Syncs changes between Firebase and Datastore

5. **Security Features**
    - Password strength validation
    - Secure session handling
    - Protection against common web vulnerabilities

Testing

To test the system:

1. **Create a test user account** through the signup page
2. **Log in** with the created credentials
3. **Navigate to the dashboard** to verify user information is correctly stored
4. **Update profile information** to test the update functionality
5. **Log out and log back in** to verify session management works correctly

Key Learning Outcomes

By implementing this system, you'll gain experience with:

1. Firebase Authentication methods and security practices
2. Google Cloud Datastore for NoSQL data storage
3. Google App Engine deployment and configuration
4. Session management and security in web applications
5. API design for authentication flows
6. Frontend-backend integration for authentication systems

The code I've provided is fully functional but would need your specific Firebase configuration details to work properly in your environment. Would you like me to elaborate on any specific part of this implementation?