Spike Doc: User Authentication Options

1. Google Sign-In

Advantages:

- Ubiquity: All of our users should have a google account through their @tamu.edu email. This also allows us to enforce a mandatory tamu.edu email domain in our account management, preventing non TAMU students from signing up and accessing the service.
- Security: Utilizes Google's security infrastructure, including two-factor authentication.
- Easy Integration: Abundant libraries and plugins available for easy integration into various platforms.
- Data Access: Allows you to access basic profile info, making onboarding faster.
- Maintenance: Google handles user management, password resets, account recovery, etc.

Disadvantages:

- Privacy Concerns: Some users might be hesitant to use their Google account due to privacy concerns.
- Dependency: Relies on Google's infrastructure. If Google experiences downtime or changes its policies, it might affect your app.
- Limited Customization: The look and feel of the login interface are controlled by Google.

2. OAuth (Facebook, Twitter, GitHub, etc.)

Advantages:

- Multiple Providers:** Users can choose their preferred platform to sign in with.
- Security: Leverages the security features of major platforms.
- Easy Integration: Like Google Sign-In, many libraries support OAuth integration.

Disadvantages:

- Scattered User Data: Managing user data across multiple platforms can be complex.
- Privacy Concerns: Similar to Google Sign-In, some users might be hesitant due to privacy issues.

- Dependency: Relying on third-party platforms can be risky in terms of stability and policy changes.

3. **JSON Web Tokens (JWT)**
   Advantages:
   - Stateless: No session data is stored server-side, reducing the server's load.
   - Versatile: Can be used in various application types, including SPAs (Single Page Applications) and mobile apps.
   - Self-contained: The token contains all the information required to authenticate the user.

   Disadvantages:
   - Storage: Tokens need to be stored securely client-side, often in cookies or local storage.
   - Token Size: Can become large if too much data is stored in them.
   - Revocation: It's challenging to revoke JWTs without introducing some state on the server side.

4. **Traditional Session-based Authentication**
   Advantages:
   - Mature: It's a well-understood model with many resources available.
   - Control: Provides more control over the login experience, UI, and session management.

   Disadvantages:
   - Scalability: Might run into issues as you scale and need to manage sessions across multiple servers.
   - Stateful: Requires the server to store session data.

5. **Biometric Authentication (Fingerprint, Face Recognition, etc.)**
   Advantages:
   - High Security: Hard to replicate or fake biometric data.
   - User Experience: Quick and often intuitive for users.

   Disadvantages:
   - Hardware Dependency: Requires users to have specific hardware features on their devices.
   - Errors: Can sometimes give false negatives or positives.

6. **Multi-Factor Authentication (MFA)**

   Advantages:

   - Enhanced Security: Even if one factor is compromised (like a password), the attacker needs the second factor to gain access.
   - Flexibility: Can use various methods as the second factor, e.g., SMS, email, authenticator apps, hardware tokens.

   Disadvantages:

   - User Experience: Adds an extra step to the login process which might deter some users.
   - Potential for Lockout: If users lose access to their second factor (like their phone), they might be locked out.

7. **Bcrypt (Ruby on Rails)**

   Advantages:

   - Secure Hashing: Bcrypt provides a strong way to hash passwords, making it computationally difficult to crack.
   - Salting: Automatically handles the creation of unique salts for each user, ensuring that even if two users have the same password, their hashes will be different.
   - Adaptive: The "cost" factor can be adjusted, allowing us to make the hashing process more computationally intensive as hardware improves, thereby increasing security.
   - Widely Adopted: Being a default choice for many Rails apps, there's a vast community and a lot of documentation available.
   - Simple Integration: Seamlessly integrates with Rails' Active Record, making it easy to add password security to your models.

   Disadvantages:

   - Speed: Bcrypt is intentionally slow to make brute-force attacks more difficult, but this might introduce latency in systems that require rapid authentication.
   - Single Use Case: It's specifically designed for hashing passwords, so it's not a full-fledged authentication solution. We'd still need mechanisms for sessions, token management, etc.