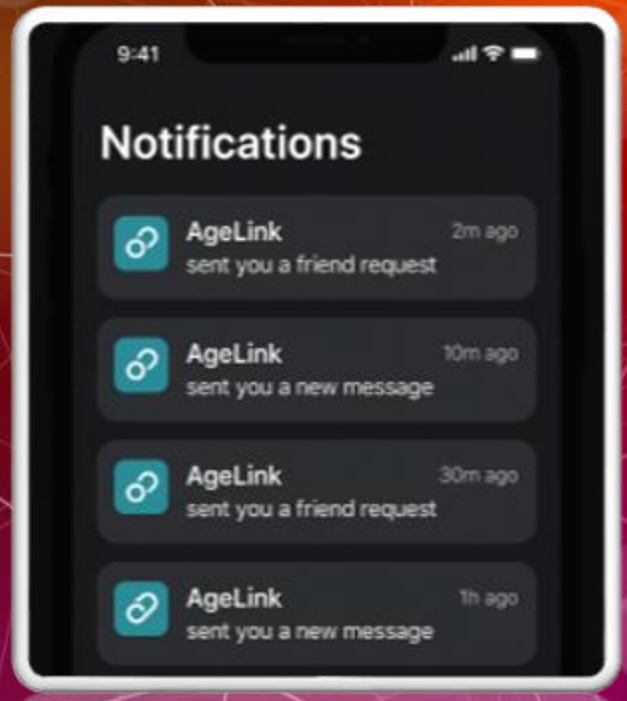
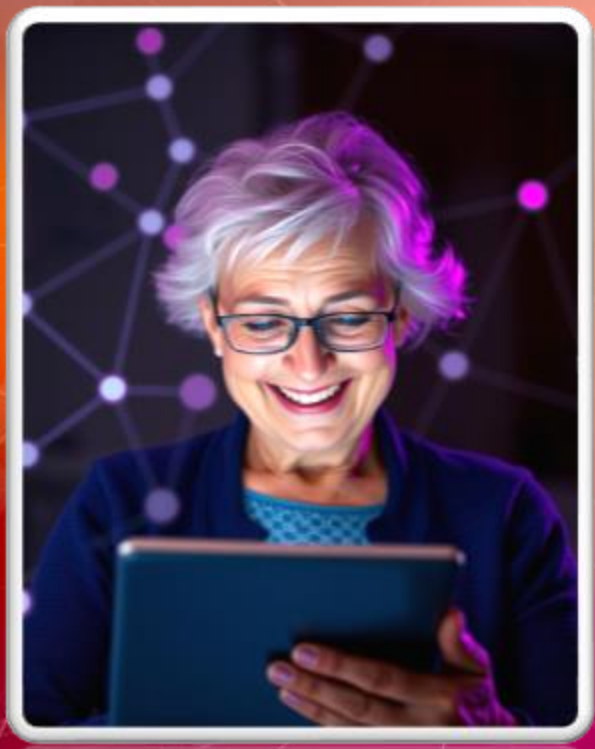
AgeLink

Connecting older adults to their communities, through simple technology

AgeLink is a lightweight, full-stack web application designed to help older adults stay connected with their local community, volunteers, and shared interest groups, whether virtually or in person. It's built to reduce social isolation and foster engagement among seniors.





Presented by – Rutvi Keshwani

What is AgeLink and What Does It Do?

As we know many seniors today feel isolated, especially those unfamilier with complex technologies. AgeLink solves this by simple web-based platform to connect to their local communities; this web-based social application is built using Python (Flask framework), HTML/CSS, and SQLite. It is designed to provide an easy-to-use online space for elderly individuals to register, view profiles, connect with others, and manage their information. It allows users to:

- Register securely
- Log in to a private dashboard
- Edit personal profiles (age, interests, short bio)
- Browse other registered users (excluding self)
- Connect with users by clicking a button next to their profile that says Connect
- View individual user profiles to learn more about them (age, bio, hobby)

The goal is to bridge social gaps for older users through a clean, user-friendly interface tailored to their needs.

Source Code & Tutorial References

During the development of AgeLink, I used several online resources and tutorials to guide my understanding and implementation of different features. Such as; link here, this helped me in understanding how to structure a flask app and login / logout functionality. Github repository link here, this taught me how to manage user sessions and protceting pages that requires protection. Db link here, I referred to this when building and querying the database (agelink.db). CSS link here, this Provided styling ideas for designing pages like profile, directory, and forms. These resources made it easier for me to learn while building AgeLink. I combined knowledge from them to create login functionality, connections, and even basic styling for a complete web experience.

What Surprised Me During Development ?

Initially, I thought this would be a simple backend-heavy project. But as I moved forward and went deeper about how I want to feature and design it I learned that; Designing for older users means prioritizing UX, larger fonts, simple buttons, and readable layout are crucial, as it is difficult for my target audience to adopt easily in their age group. I realized how important it is to improve any minor UI such as; spacing, colors, feedback messages, as it dramatically improved the experience. When I tried testing with basic CSS language it taught me how visual feedback plays avital role as a backend validation, such as: Profile updated!. Most importantly I learned; Good design isn't optional, it's essential, especially for a non-technical audience (target audience).

Changes to Planned Features

When I first planned AgeLink, it was supposed to be a simple registration and profile app. But as the project developed, I realized it needed to be more interactive and user-friendly.

My original plan:

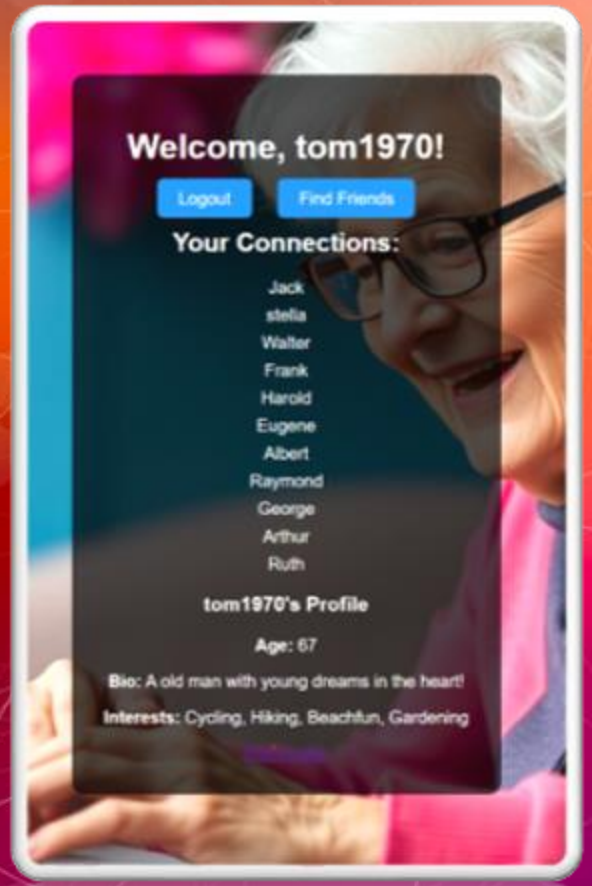
- Basic login, registration
- Only showed user's own info
- Static content layout

Key changes I made:

- Edit Profile page added to let users update their age, bio, and interests
- User Directory shows all registered users (excluding the logged-in user)
- Connect Button added to let users connect with others
- Users can now view other users' profiles by clicking their names
- Feedback messages like "Profile Updated" or "Invalid Password" added for better UX
- Included Back to Profile or Back to Home buttons on each page for easier navigation

These changes made AgeLink feel more like a real social platform and much easier to use.

Future Features: Also Planning to add messaging feature using Socket.IO from Javascript Library.



Major Roadblocks & How We Solved Them

As I was moving forward in the development of AgeLink I decided to add some new features as I mentioned in previous slid, while working on those new features, I faced few challenges that taught me a lot.

Here are the two major roadblocks I faced and how I handled them:

1. Flash messages weren't showing

Issue: upon inserting a wrong / invalid user or password, there was no error message popoing up.

Solution: Added get_flashed_messages() block to HTML templates

Ensured user actions (e.g., "Login failed", "Profile updated") had clear feedback

2. Profile updates not saving

Issue: When trying to update age, bio, or interests, the changes weren't being saved properly.

Solution: I debugged the update_profile() function and corrected a missing argument that was causing it to fail.


Thank you!