Sprint 1 Release

1 Research Hosting

Description:
Survey a number of hosting platforms and determine which one(s) is/are fit for this project.

Acceptance Criteria:

- The hosting platform should be easily integrated with the project.
- The hosting platform should be cost effective.

Tasks:

- Evaluate a number of ways to host the project.
- Determine which is/are best fit.

Evidence of Completion:

- There is a team consensus on how we will host our project.

Assigned to: Connor

Outcome: ✅ Complete

---

12 Research Social Media Implementation

Description:
Research common social media application features and architectures to inform design decisions for the project.

Acceptance Criteria:

- At least two social media platforms or implementations are researched.
- Key features and patterns are identified and documented.

- Findings are relevant to the project's goals and scope.

Tasks:

- Identify comparable social media platforms.
- Research core features (e.g., feeds, profiles, interactions).
- Document findings and architectural insights.

Evidence of Completion:

- Written summary or document outlining researched platforms and key takeaways.

Assigned to: All

Outcome: 🔃 In-Progress Throughout All Sprints

---

14 Discuss Features

Description:
Collaboratively discuss and define the initial feature set for the application based on project goals and research findings.

Acceptance Criteria:

- Core features are clearly identified and prioritized.
- Feature scope is appropriate for the project timeline.
- Decisions are documented and agreed upon by the team.

Tasks:

- Review research findings and project requirements.
- Brainstorm and prioritize potential features.
- Document finalized feature list.

Evidence of Completion:

- Documented list of agreed-upon features with brief descriptions.

Assigned to: All

Outcome: 🔃 In-Progress Throughout All Sprints

---

2 Create Wireframes/Models of UI

Description:
Create wire frames for each page, showing precisely where everything will be placed.

Acceptance Criteria:

- There is a wireframe for each page
- If a page has different variations, there are wireframes for the variations
- The wireframes show all buttons, textboxes, and position of each element of each page

Tasks:

- Create a wireframe for each page/page variation
- Decide where elements will go, and represent them with placeholder graphics

Evidence of Completion:

- The wireframes accurately represent each aspect of the website, are exported as PNGs, and are added to the documentation files.

Assigned to: Matthew Eagleman

Outcome: ✅ Complete

---

3 Create a Database Model

Description

Design a conceptual database model that represents the core entities and relationships of the Passerby social media application.

User Story

As a developer, I want a clear database model so that the backend can be implemented consistently and correctly.

Acceptance Criteria

- An ER diagram exists using standard ER (Chen) notation
- The model includes entities for users, friendships, posts, messaging, and groups
- Relationships and attributes accurately reflect the friends-only social media design

Tasks

- Identify core entities and relationships required by the application
- Design an ER diagram representing these entities and relationships
- Review the model for consistency with Sprint 1 features and requirements

Evidence of Completion

- A finalized ER diagram is included in the Architecture Document
- The diagram clearly shows entities, attributes, primary keys, and relationships

Testing / Verification

- A reviewer can trace each major system feature (users, friendships, posts, messages, groups) to an entity or relationship in the ER diagram

Assigned to: Jacob

Outcome: ⏩ Deferred to Sprint 2

---

4 Create a User Data Model

Description:
In the backend, create a model/entity that represents a user of the application and all of their attributes (username, password, email, profile picture, etc.).

User Story:
As a user, I want to store essential information about my account so that Passerby understands who I am.

Acceptance Criteria:

- There will be a database (a table most likely) from which information about a user may be easily retrieved. Attributes should include: user_id, username, password, email, profile picture, bio, and status. Each row represents a single user with a unique ID.

- The data model should keep private information like passwords hashed.

- User models must be able to be easily added to and deleted from the database.

Tasks:

- Create a secure schema that represents a user.
- Make it so that each user has a unique identifier and a hashed password.

Evidence of Completion:

- The application can easily retrieve information about a user (user_id, password, etc.) that has been stored securely in the backend.

Commit:
- TBD

Testing / Verification:

- A user can successfully create a data model
- The user's data is stored securely

Assigned to: Owen

Outcome: ✅ Complete

---

23 Implement Login / Sign up

Description: Implement authentication so users can sign up and log in to Passerby. The app uses a single main page (feed or profile); if the user is not authenticated, the feed remains the route, but an auth UI is shown on top of it (modal/panel)

User Story: As a user, I want to create an account or log in so that I can access the Passerby feed with my identity and saved profile information

Acceptance Criteria:

- Users can sign up using a username, email, and password.
- Users can log in using email + password (or username + password), and receive a valid authenticated session/token
- If a user is not logged in, the app still loads the feed page route, but displays an inline auth experience (modal/overlay/panel) instead of navigating to a separate /login page
- On successful login/sign up:
    - The auth overlay disappears and the user can interact with the feed immediately.
    - Passwords are never stored in plaintext; they are hashed in the backend
- The backend returns clear success/failure responses:
    - Correct credentials → authenticated
    - Incorrect credentials → error message
    - Duplicate email/username on sign up → error message
    - Users can log out, which returns them to the “feed route + auth overlay” state

Tasks:

- Build backend endpoints/services:
    - POST /auth/signup (creates user, hashes password, returns session/token)
    - POST /auth/login (verifies password hash, returns session/token)
    - POST /auth/logout (invalidates session/token if applicable)
    - (Optional) GET /auth/me (returns current user from session/token)
- Implement password hashing (e.g., bcrypt/argon2) and verification
- Implement session management:
    - Either JWT (stored securely) or server sessions (cookie-based)
- Frontend:
    - Create a single-page auth UI component that can toggle Login ↔ Sign up.

    - Show auth UI when user == null (not authenticated), while still rendering the feed in the background
    - On auth success, set user state and hide overlay
    - Add a logout button/action that clears session state
- Add basic validation + user-friendly errors (missing fields, weak password, invalid credentials)

Evidence of Completion:

- A new user can successfully sign up; their user row is created in the database
- A returning user can log in successfully with the correct credentials
- Passwords in the database appear hashed
- When logged out, visiting the app shows the feed route but with the auth overlay/panel instead of redirecting to a /login
- The app can retrieve the authenticated user's info after login

Commit:
- TBD

Testing / Verification:

- Sign up with a new account → success; verify user exists in DB; verify password is hashed.
- Sign up with an existing email/username → correct error returned.
- Log in with correct credentials → success; session/token present; overlay disappears.
- Log in with wrong password → correct error returned.
- Refresh page while logged in → still authenticated (session persists).
- Log out → session cleared; auth overlay returns on the feed route.

Assigned to: Bryson (Connor - Assist)

Outcome: ✅ Complete

---

6 Login/Create User

Description:
Users should be able to set a username and a password as the required credentials to access their profile.

User Story:
As a user, I want to be able to create a profile using a username and password so that I can access my account when I want to.

Acceptance Criteria:

- The user must enter their username, password, and email to sign up for an account

- The user's credentials shall be stored in a database for easy retrieval.
- Incorrect username/password combinations will deny access and display a notification prompting the user to try again.
- An option will be given to the user to reset their password if they forgot it. Password recovery must be sent via the email provided during sign-up.

Tasks:

- Let the user sign up/log in by entering information into a text box.

- Make it so that the frontend of the application can communicate with the database tht stores the user's information
- Implement password recovery

Commit:
- TBD

Assigned to: Owen

Outcome: ✅ Complete

---

8 Save User Data

Description:
Securely store the user's account information whenever they create/edit their account.

Acceptance Criteria:

- The user's information will be safely stored in a database in the backend.

- Information about the user must be able to be easily retrieved after it is stored.

Tasks:

- Build a database that can store all of the relevant information about a user.

- Implement security features like hashing passwords.

Commit:
- TBD

Assigned to: Owen

Outcome: ✅ Complete

---

13 Set up GitHub

Description:
Create a GitHub repository for the project.

Acceptance Criteria:

- A public GitHub repository is created for the project.
- All team members have access as Contributors.
- There are branch protections requiring pull requests to update the main branch.

Tasks:

- Create the GitHub repository.
- Invite all team members as contributors.
- Add branch protections to the main branch.

Evidence of Completion:

- There is a public Github repository that all team members have contributor access to, and it has protections on the main branch.

Assigned to: Connor

Outcome: ✅ Complete

---

16 Implement Dashboard

Description

Create the main dashboard page that users see after logging in, serving as the central hub of the application.

User Story

As a user, I want a dashboard so that I can access my feed, profile, and other core features from one place.

Acceptance Criteria

- A dashboard page component exists in the frontend
- The dashboard renders successfully after authentication
- Placeholder content is acceptable for Sprint 1

Tasks

- Create a dashboard page/component in the React frontend
- Add placeholder UI elements representing feed and navigation areas
- Ensure the dashboard integrates with existing routing

Evidence of Completion

- The dashboard page renders in the browser without errors
- Users can navigate to the dashboard after logging in

Commit

- Add dashboard page component

Testing / Verification

- Log in to the application → dashboard page loads correctly
- No runtime or rendering errors appear in the console

Assigned to: Jacob

Outcome: ⏩ Deferred to Sprint 2

---

17 Implement Profile Page

Description

Create a user profile page that displays basic user information.

User Story

As a user, I want to view my profile so that I can see my personal information within the application.

Acceptance Criteria

- A profile page component exists
- The profile page displays user-related fields (username, bio, profile picture placeholders)
- The page renders without errors

Tasks

- Create a profile page component in the frontend
- Display placeholder or retrieved user information
- Integrate profile page into application navigation

Evidence of Completion

- Profile page renders successfully in the browser

- Navigation to the profile page works correctly

Commit

- Add profile page component

Testing / Verification

- Navigate to the profile page → correct page loads
- Profile content is visible and formatted correctly

Assigned to: Jacob

Outcome: ⏩ Deferred to Sprint 2

---

19 Implement Page Navigation

Description:
Implement basic client-side navigation within the React application to allow users to move between core pages without full page reloads.

User Story:
As a user, I want to navigate between pages smoothly so that I can access different parts of the application efficiently.

Acceptance Criteria:

- Users can navigate between at least two pages (e.g., App and About Page).
- Navigation occurs without a full browser reload.
- URLs update correctly when navigating between pages.

Tasks:

- Install and configure React Router.
- Create placeholder pages for navigation testing.
- Implement navigation buttons (RouteButton component).

Evidence of Completion:

- Working navigation demonstrated in the browser with correct URL changes.

Commits:
- `Add router`

- `Add basic routing capabilities`

Testing / Verification:

- Manually verify navigation by clicking links and confirming page content updates correctly.

Assigned to: Connor

Outcome: ✅ Complete

---

20 Create Initial Database

Description:
Set up the initial Supabase project and invite all team members to have access.

Acceptance Criteria:

- A Supabase project is created and configured.
- At least one table exists with appropriate fields and primary key.
- All team members have access to the Supabase project.

Tasks:

- Create the Supabase project.
- Define initial database test table and columns.
- Invite all team members to have access.

Assigned to: Connor

Outcome: ✅ Complete

---

21 Connect Frontend to Database

Description:
Connect the React frontend to the Supabase backend to enable data retrieval and interaction.

User Story:
As a user, I want the application to load real data so that information is persistent and dynamic.

Acceptance Criteria:

- Supabase client is configured in the frontend.
- Frontend can successfully fetch data from the database.
- No connection or authentication errors occur.

Tasks:

- Install Supabase JavaScript client.
- Configure environment variables.
- Implement a test database query in the frontend.

Evidence of Completion:

- Data from the database is displayed in the frontend UI.

Commit:
- `Test connection with Supabase`

Testing / Verification:

- Confirm data loads correctly by refreshing the page and checking the browser console for errors.

Assigned to: Connor

Outcome: ✅ Complete

---

22 Create Initial Frontend

Description:
Create the initial React frontend using Vite.

Acceptance Criteria:

- React + Vite project is created and runs locally.
- Application builds without errors.

Tasks:

- Initialize Vite React project.
- Configure project structure and dependencies.
- Create initial components and layout.

Evidence of Completion:

- Frontend loads successfully in the browser with placeholder content.

Commit:
- `Initial Commit using React and Vite`

Testing / Verification:

- Run the development server and verify the UI renders without errors.

Assigned to: Connor

Outcome: ✅ Complete

---

24 Create Navigational Diagram

Description:
Create a diagram showing rough page layouts, and arrows showing how you get from one page to another.

Acceptance Criteria:

- The diagram communicates how to get from one page to another effectively
- All pages are reachable

Tasks:

- For each page draw a rough layout, with any buttons that connect to other pages
- Draw arrows from the button you need to push, to the page that it leads to

Evidence of Completion:

- The diagram is made, readable, and added to the Architecture document

Assigned to: Matthew Eagleman

Outcome: ✅ Complete