Scalable Angular Clients folder structure

Context and Problem Statement

We currently have a very fractured folder structure in the Angular clients with multiple competing folder structures. We need to standardize on a single folder structure which will allow us to continue growing without friction. This ADR builds upon 0010 Use Angular Modules and proposes a folder structure.

Resources

This is heavily based on the following resources.

- Angular - NgModules
- Angular - Application structure and NgModules
- Angular - Lazy-loading feature modules
- Using Nx at Enterprises

Considered Options

- **Nx folder structure** - Seems like a well thought out structure but it essentially requires that we use nx since it relies heavily on npm packages. We're not yet at a stage where we feel comfortable adopting such a tool.
- **Lightweight structure inspired by Angular Docs** - Takes inspiration from the Angular docs + Nx to propose a more lightweight structure that still keeps the features in the `app` directory.

`SharedModule` & `CoreModule`

The Angular docs specifies that there should be a `SharedModule`. It further specifies that the shared module should not provide providers. A common convention is to create a `CoreModule` which is responsible for setting up the core providers.

Proposed Folder structure

This folder structure is based on our existing routing structure, since a common pattern is to use modules for nested routes in order to support lazy loading. The root folders are mostly based on the root routing concepts we have, with various public routes being categorized under `accounts`.

```
web/src/app
  core
    services
    core.module.ts
    index.ts
  shared
    pipes
    components
    shared.module.ts
    index.ts
  accounts
  providers
  reports
  sends
  settings
  tools
  vault
    shared
      vault-shared.module.ts (this gets imported by Organization Vaults)
    vault.module.ts
    index.ts (exposes the following files:)
      vault-shared.module.ts
      vault.module.ts
  --
  app.component.html
  app.component.ts
  app.module
  oss-routing.module.ts
  oss.module.ts
  wildcard-routing.module.ts
```

This structure will be implemented in multiple steps.

1. Extract unrelated files from `src/app`. https://github.com/bitwarden/clients/pull/3127
2. Create `CoreModule`. https://github.com/bitwarden/clients/pull/3149
3. Create `SharedModule`. https://github.com/bitwarden/clients/pull/3222
4. Migrate all existing loose components to `SharedModule`.
   - Any remaining functionality in root should provide a module.

Decision Outcome

Implementing a file structure modeled after the above example.