# Plan to Adapt Team Features for Individual Users

## Overview of Current Team-Based System

The existing implementation is structured around **teams**. Key components include a teams database table, server-side logic for team operations, API endpoints for team management, and front-end UI elements for team creation/joining and progress tracking. The code is well-organized, separating the database schema, storage (business logic), API routes, and UI:

* **Database Schema (shared/schema.ts):** Defines a teams table with fields such as unique code, name, currentPhase, cohortTag (grouping identifier), accessToken for auth, avatarIcon, timestamps, etc[[1]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L7-L15). It also defines related tables like phaseData (to store each team's work per phase) and cohorts (to group teams)[[2]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L22-L30)[[3]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L39-L47). Validation schemas (using Zod) exist for team creation, login, updates, etc[[4]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L96-L104)[[5]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L108-L116).
* **Storage Layer (server/storage.ts):** Implements business logic for team operations. For example, it provides methods to create a team, get teams by code or ID, update a team’s phase or avatar, assign teams to a cohort, list teams, etc[[6]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L9-L17)[[7]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L16-L20). It also handles saving and retrieving phase data for a team (so progress can be resumed)[[8]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L211-L219). Notably, createTeam auto-assigns a random avatar and generates an accessToken (if not provided) for the new team[[9]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L76-L84)[[10]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L80-L88).
* **API Routes (server/routes.ts):** Exposes HTTP endpoints for team functionality. Key routes include:
* **Team Auth:** /api/auth/team/login (logs in a team using an access token) and /api/auth/team/logout (clears the session)[[11]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L103-L111)[[12]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L114-L122), plus a status check.
* **Team Management:** /api/teams (POST to create a team, GET to list all teams)[[13]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L199-L207), /api/teams/:code (GET to retrieve a team by its code)[[14]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L187-L195), /api/teams/:id/phase (PATCH to update team’s current phase)[[15]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L208-L216), /api/teams/:id/avatar (PATCH to change team avatar)[[16]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L222-L231)[[17]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L232-L240), and /api/teams/:teamId/website (PATCH to submit a website URL for the team’s project)[[18]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L306-L315)[[19]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L319-L325).
* **Admin & Cohort:** Endpoints for assigning/unassigning teams to cohorts and viewing cohort status. For example, admins can GET /api/admin/cohorts/:cohortTag/teams\_status to see all teams in a cohort with their submission and voting status[[20]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L481-L489). There are also endpoints for voting and viewing results in a cohort (teams vote for others’ submissions)[[21]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L499-L508)[[22]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L513-L522).
* **Front-End Components:** The client app uses React (TypeScript) components to handle team interactions:
* TeamModal.tsx: Interface for creating a new team or joining an existing team (likely by entering a team code and name).
* NavigationHeader.tsx: Displays current team info (name, avatar, etc.) and provides navigation (possibly team switching or logout).
* AvatarSelector.tsx: Allows choosing a team avatar from preset icons.
* Phase.tsx (or PhasePage): Renders content for the current phase of the team’s project, aware of the team context.
* Admin.tsx: An admin dashboard showing teams, cohorts, and controls for instructors.
* App.tsx: Sets up team authentication on app load (e.g. checking local storage for an access token and verifying session) to keep a team logged in across refreshes.

This team-based system allows a group of users (team members) to share a code to access a common workspace. Progress is tracked per team (not per individual), and instructors (admins) can oversee all teams in a cohort. **The goal is to adapt this architecture for individual users**: each person will use the platform independently (no multi-member teams), but we want to **preserve the core functionality** – persistent sessions with a code, progress saving, multi-device resume, and instructor oversight – while removing or refactoring the “team” grouping aspect.

## Adapting the System for Individual Use

Under the individual model, each user effectively becomes their own “team of one.” We can leverage most of the existing structure by treating an individual’s session as a team internally. Below is a comprehensive plan, covering necessary changes and recommendations across the stack:

### 1. Database Schema and Data Model

* **Repurpose or Rename the Teams Table:** We can continue using the teams table to represent individual participants. Each row will correspond to one user’s session (instead of a collaborative team). The schema already supports fields that make sense for individuals:
* code: a unique access code for the individual’s session (they will use this to resume work on other devices, just like teams did)[[23]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L7-L14).
* name: we can use this to store the individual's name or a chosen nickname for their session (previously this was a team name)[[23]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L7-L14).
* currentPhase: tracks the user's current section or phase in the curriculum[[24]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L9-L17).
* avatarIcon: the user’s avatar image identifier[[25]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L13-L17).
* cohortTag: group label to assign the individual to a cohort (e.g. a class or course batch)[[26]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L11-L19).
* submittedWebsiteUrl: link to the user’s project submission (if applicable, e.g. final project URL)[[27]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L10-L17).
* accessToken: a unique token for session authentication[[28]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L12-L15).

We might consider **renaming** this table to something like participants or user\_sessions for clarity, but that would involve updating references throughout the code. If a quick adaptation is preferred, we can keep the table named "teams" in the code and simply treat it as the participant store. The important part is that each “team” entry now represents one individual.

* **No User Accounts for Participants:** The system remains **login-free for learners**, which is consistent with the current design. In the existing schema, the users table is only used for admin accounts (with username, password, role)[[29]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L32-L37). We will keep it that way – participants will not need accounts, they will continue to use the code-based access. This means **no changes needed to the users table** except perhaps ensuring an admin or instructor account exists. Each participant’s identity in the system is just their entry in the (now individual) team table plus their code.
* **Validation Schemas:** The Zod schemas for team creation and updates can be reused almost as-is. For example, insertTeamSchema currently requires a name and code (and auto-fills missing fields like avatar or token)[[30]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L64-L72). We should still validate that an individual provides a name (if we require one) and ensure the code is unique. We might want to adjust validation messages to refer to “session” or “participant” instead of team, but functionally the schemas are fine. The same goes for update schemas like updateTeamPhaseSchema, updateTeamAvatarSchema, etc., which we can reuse to validate individual progress updates[[31]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L100-L108)[[32]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L104-L112).
* **Cohorts and Other Tables:** The cohorts table can remain unchanged[[3]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L39-L47). It will now group individuals instead of teams, but conceptually it’s the same (each cohort might be an instance of the course or a class grouping professionals in a training). The votes table (if the voting feature is retained) likewise stays the same structurally[[33]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L51-L59) – individuals will vote as themselves (each vote record references a “voting team” and “voted for team” by ID[[34]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L52-L60), which in individual mode are just user entries voting for each other). If the context of the training includes a showcase or peer voting on final projects, this can still function with individuals. If not needed, we could disable or remove the voting feature, but since the question suggests keeping core functionality, we’ll assume it stays. The phaseData table (storing JSON data for each phase of a team’s progress) is perfectly suited to individual work – each user’s answers or work in each phase is stored keyed by their team (user) ID and phase number[[2]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L22-L30). No changes needed there, as it already works per team/phase.

**In summary**, we will reuse the team-based schema for individuals. Each participant gets a unique code and entry in teams. We should update naming in documentation and code comments to avoid confusion (e.g., clarify that Team now represents an individual session). Actual schema changes are minimal: if anything, we might remove fields that no longer make sense, but all the existing fields still seem applicable to individuals. (For instance, name, avatarIcon still apply. If “team name” was meant to be a project name, we can treat it as either the person’s name or allow them to title their project.)

One potential addition: if we want to avoid *code collisions* in a scenario where each individual picks their own code, we might choose to **auto-generate codes** instead of relying on user-provided codes. In the current system, when creating a team, the code is provided by the request and must be unique (the API returns an error if it conflicts)[[35]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L176-L184). With many individuals, the chance of duplicate code entries or users picking simple codes increases. To ensure uniqueness and reduce user effort, generating a random code (or using a combination of words) for each new session could be wise. This would require removing the code from user input in the create flow and generating it server-side (we could repurpose the existing code generation logic used for access tokens, or simply use a GUID/short ID). This is an optional improvement – the current approach of user-chosen codes with uniqueness check will still work for individuals.

### 2. Backend Storage Layer Adaptations

The storage layer (DatabaseStorage class in server/storage.ts) already encapsulates all necessary operations for teams, which we can largely keep the same for individuals. Key functions and how to adapt them:

* **Creating a Session (Team):** The createTeam function will now create an individual session. It currently inserts a new team with a random avatar and generates an accessToken if one isn’t supplied[[10]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L80-L88). It also uses a provided code and name. We might want to adjust how codes are handled (as mentioned, possibly auto-generate). Otherwise, this logic is fine – when a participant starts a new session, we call createTeam with their name and chosen code, and the storage layer will assign them an avatar and token. The return is a Team object (which in our context is now a participant record).
* **Lookup Functions:** Functions like getTeamByCode, getTeamById, and getTeamByAccessToken remain essential[[36]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L93-L101)[[37]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L102-L105). They allow retrieving an individual’s record either by the code they entered or by an auth token. For example, when someone tries to resume using a code, the app can find their ID and token via getTeamByCode. We should continue to use these for login flows (the login route uses getTeamByAccessToken specifically)[[38]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L106-L114)[[39]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L118-L125).
* **Phase Progress Updates:** updateTeamPhase(id, currentPhase) updates the currentPhase field in the DB for a team[[40]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L108-L116). We’ll use this to track an individual’s progress through the training. In a sequential flow, this increments as they move section to section. If the course remains linear, we keep using this (e.g., after finishing Phase 1, call updateTeamPhase to set currentPhase=2). If the sections are truly “atomized” (meaning the participant can do them in any order), we might repurpose this field or deemphasize it. For instance, we could interpret currentPhase as the highest completed section or simply not rely on it heavily (and allow accessing sections out of order via direct links). However, given the existing design, it’s likely the training is meant to be done in order, so currentPhase is useful to gate progress. We will maintain it to prevent skipping ahead unless desired otherwise.
* **Phase Data Storage:** The functions to save and retrieve phase data (savePhaseData, getPhaseData, etc.) continue to work per individual. No changes required – they index by teamId (user session ID) and phase number[[8]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L211-L219)[[41]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L219-L224). This is how the user’s inputs or results from each phase are saved so they can leave and come back later. We will use these as-is. If anything, we might add convenience methods in the future to get an overview of all phase completions for an individual, but getAllPhaseDataForTeam already provides all records for a team[[41]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L219-L224).
* **Avatar Updates:** The updateTeamAvatar function sets a new avatar icon for a team (user)[[42]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L130-L138). This remains useful for letting individuals change their avatar. The logic doesn’t change – it’s just updating a single field.
* **Website Submission:** updateTeamWebsite saves a URL (or null) for the team’s project[[43]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L120-L128). We keep this for individuals to submit their final project link. The validation and formatting (ensuring proper URL, adding https:// if missing) is handled by the schema and route[[5]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L108-L116)[[44]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L314-L322), which we’ll leave in place.
* **Cohort Management:** The storage has assignTeamsToCohort(teamIds[], cohortTag) and corresponding unassign function[[45]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L144-L152)[[46]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L160-L167). These will be used to group individual participants into cohorts. We can reuse them directly – for example, when an instructor wants to put certain users into the “Spring 2025” cohort, the backend will call assignTeamsToCohort with their IDs and the cohort tag. Even though the name says “Teams,” it will just handle arrays of single-person team IDs. We might later rename these methods to assignParticipantsToCohort for clarity, but functionally no change is needed. Similarly, getTeamsByCohort(cohortTag) will list all participants in that cohort[[47]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L174-L182), which is exactly what an instructor interface would need.
* **Voting:** If we preserve peer voting, storage methods like submitVotes and getVotesByTeam/getVotesByCohort remain relevant[[48]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L314-L322)[[49]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L320-L328). Each individual can vote for others. No changes needed in these methods; they already ensure things like no self-voting by design (enforced via a DB constraint and also checked in code)[[50]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L56-L62)[[51]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L564-L572). If the training scenario changes such that voting isn’t needed (for example, if these professionals are not voting on each other’s work), we could disable or remove the voting endpoints. But since you indicated keeping core functionality, we’ll assume this stays available for any collaborative or competitive activities in the course.

**Backend logic summary:** The existing storage functions cover all needed operations. We mostly **reuse them unchanged**, while conceptually thinking of each “team” as an individual. We should audit the code to remove any assumptions of multiple members. For instance, there is no function in storage for adding a user to a team (since no user accounts are used for team membership), so we’re fine on that front. The only consideration is perhaps enhancing some functions for individual usage, such as: - Ensuring getAllTeams() (which lists all teams) can handle a potentially larger number of entries (if a cohort has many individual participants, this could return more records than when teams of several people were grouped – if needed, we could implement pagination or filtering, though not strictly required if numbers are still moderate). - Possibly adding a new method to get an overview of an individual’s progress (though this can be derived from currentPhase and presence of phaseData entries, or via the admin endpoints as discussed below).

Overall, the backend remains robust for our needs with minimal tweaks. We’ll carry forward the separation of concerns (database logic in storage, accessed by routes), which will make these changes easier to manage.

### 3. API Route Adjustments

The HTTP API will continue to function similarly, but we will adjust terminology and tighten behaviors for individual use:

* **Session (Team) Authentication:** We will keep using the /api/auth/team/login and /api/auth/team/logout endpoints, but these effectively become **session login/logout** for individuals. Currently, a client logs in by sending an access\_token (which encodes their team identity) to /api/auth/team/login, and the server looks up the corresponding team and establishes a session[[38]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L106-L114)[[52]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L114-L123). This flow remains ideal: an individual provides their saved access token (from local storage) or goes through the code exchange to get it, and we start a session. We might rename these endpoints to /api/auth/session/login for clarity when we update the client, but it's not strictly necessary to change the route paths if doing so would require widespread changes. The login route’s logic doesn’t need changes; it already checks the token and creates a session (setting req.session.teamId internally)[[12]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L114-L122). For logout, it destroys the session as before[[53]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L135-L141).

One improvement to consider is **direct code-based login**: currently, the client likely calls /api/teams/:code to get the team’s access token then calls /api/auth/team/login. We could streamline this by having a single endpoint where a user enters their code and the server both validates it and logs them in (for example, a new endpoint /api/auth/code-login that takes a code, finds the team, and sets the session in one step). This would remove one round-trip. However, this is an enhancement – the existing two-step process (code -> get token -> login) works and can be reused for now.

* **Team Creation -> Session Creation:** The POST /api/teams endpoint is currently used to create a new team[[54]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L183-L191)[[55]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L163-L171). We will use this for individuals starting a new session. Changes to consider:
* Input: originally expects a name and code (and perhaps other optional fields like avatar, though avatar is auto-assigned)[[56]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L161-L169). For individuals, we’ll ask for perhaps just a name (their name or a nickname) and possibly let the code be auto-generated or still user-chosen. The backend will uppercase and trim the code and ensure uniqueness as it does now[[57]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L164-L172).
* Behavior: After parsing input and creating the new team (individual), the route currently returns the full team object including the generated accessToken[[58]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L170-L175). On the client side, we should then immediately use that token to log the user in (or we could automatically set the session server-side). In the current flow, it likely just returns the team and the client calls the login endpoint. We can keep that flow.
* We should **change response or usage if needed**: one possible tweak is to not expose the raw accessToken in the JSON response for security. Right now, if you create a team, the entire team record is JSON-ed to the client, which would include the token and could even include other fields. This is convenient but slightly risky if someone were watching network traffic. However, since the client needs the token to authenticate, it’s a necessary disclosure. We can mitigate by using HTTPS and/or by scoping what we return (maybe omit the token and handle login differently). But given a time-constrained adaptation, it’s acceptable to keep as is.
* **Fetching by Code:** GET /api/teams/:code returns the team record if it exists[[14]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L187-L195). This was likely used to let a second team member join an existing team by retrieving the team info (and token) via the known code. In individual mode, **sharing codes isn’t needed** except for the user retrieving their own session. But we can still allow this endpoint so that if a user comes with just their code (and not the token), the client can fetch their record. Essentially, this becomes a “resume session by code” lookup. We should consider tightening access: as it stands, this endpoint is public (no auth) and will return any team’s data to anyone who knows the code. In a team scenario, codes were presumably shared among intended members only. With individuals, a code should be kept private by the user (like a password). It might be unlikely someone guesses another’s code, but if codes are short or user-chosen, it’s a potential issue. We might improve security by requiring an extra verification (perhaps the user’s name or a simple passphrase) to fetch the data, or by not returning the full record (especially not the token) without authentication. However, since the whole system is designed around easy access with just a code, we’ll likely keep this behavior for now. We should at least ensure users understand to keep their code secret. If we wanted, we could remove this endpoint and instead have code login as described earlier. For the immediate adaptation, keep it: it will allow a participant to retrieve their session info on a new device by providing the code.
* **Updating Phase and Data:** The endpoints for updating a team’s current phase (PATCH /api/teams/:id/phase) and for saving phase data (POST /api/phase-data) remain appropriate. When an individual completes a section, the frontend can hit these to record progress. These are not protected by auth in the current code (meaning if someone knew your team ID, they could in theory update your phase or read your phase data). It might be prudent to add authentication here (only the logged-in session’s team should modify their data). In a single-user scenario, this is less of an issue if codes are secret, but still, an improvement could be to use ensureAuthenticatedTeam middleware on these routes so only the session owner can manipulate their data. Currently, ensureAuthenticatedTeam is used for voting and status endpoints[[59]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L52-L59)[[60]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L513-L521), but not for the general phase data routes. For better security, we might add it, but doing so means the client must have a session established (which they will if they logged in properly). We should test these flows when implementing to ensure we don’t inadvertently block legitimate uses. In short: **keep the phase data endpoints** and consider requiring auth for them.
* **Avatar and Website Endpoints:** These are fine to keep as is:
* PATCH /api/teams/:id/avatar: individuals can call this (likely via the UI) to update their avatar. It currently doesn’t require auth in code, but we might also protect it with ensureAuthenticatedTeam or at least verify on the server-side that the session’s teamId matches the ID being patched. In the existing code, they do verify the team exists but not that the caller is that team[[61]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L226-L234). For an individual setup, it might be okay (the worst someone could do is change another’s avatar if they guessed an ID). It’s a minor risk but easy to fix by limiting access. We could add a check: if (req.session.teamId !== teamId) then forbid the change (except for admin).
* PATCH /api/teams/:teamId/website: used to save the project URL. Similarly, we should ensure only the correct user (or an admin) hits this for their own ID. The route currently just processes the URL and updates the team entry[[18]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L306-L315)[[19]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L319-L325). We keep the functionality (so individuals can submit their final work link). Just adjust any messaging (the client UI might say “Submit your team’s website” – we’ll reword that to “Submit your project URL” or similar).
* **Admin and Instructor Endpoints:** All admin routes will continue to operate, but now they apply to participants. We absolutely **keep instructor visibility features**:
* /api/admin/cohorts/:tag/teams\_status – this returns a list of teams (participants) in a cohort with some flags[[20]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L481-L489). Currently it adds hasSubmitted and hasVoted for each team. We will maintain this, and possibly extend it. Since in an individual context the instructor might want to quickly see progress, we could modify this endpoint to also include each participant’s currentPhase or percentage completion. The data is readily available (the team object has currentPhase field[[39]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L118-L125)). We can include it in the response so the front-end can display how far along each person is. This would be a helpful addition to instructor oversight: for example, an instructor could see that “Alice (Code XYZ) is on Phase 3, has not submitted the final project yet, and has not voted” all in one view.
* /api/admin/assign and /unassign teams to cohorts – unchanged except in usage. The instructor will use these to manage participants grouping. Perhaps adjust front-end wording from “Assign Teams” to “Assign Participants” but the API endpoints and storage functions remain the same[[45]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L144-L152)[[62]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L160-L168).
* Admin auth (/api/auth/admin/login etc.) remains exactly as is – instructors will log in with their username/password as they do now[[63]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L62-L71)[[64]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L88-L96). No changes needed there.
* We may want to add admin APIs to inspect individual progress in detail. For instance, an endpoint like /api/admin/team/:id/phase-data could allow an instructor to fetch all phase data for a given participant (essentially calling getAllPhaseDataForTeam). Right now, the admin could retrieve some of this via multiple calls or even using the existing endpoints if they know the team ID, but a dedicated admin helper could streamline it. This is an optional enhancement. It would allow the instructor to review a participant’s answers or work for each section, which could be useful for grading or feedback in the training context. If implementing this, we’d protect it with ensureAuthenticatedAdmin and use storage.getAllPhaseDataForTeam(id)[[65]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L268-L275) to return the data. The front-end Admin page could then have a UI to view details when an instructor clicks on a participant.
* **Route Naming:** In the short term, we might leave the routes as /api/teams/... for minimal backend changes. The client can still call them; the concept of “team” is just an internal detail. If desired, we could set up aliases (e.g., make /api/participants routes that mirror the team ones) or fully rename. Renaming means adjusting the client API calls (which is doable since we’ll be editing the front-end components anyway). For clarity, an eventual rename might be:
* /api/session or /api/participant instead of /api/team in paths and payloads.
* For example, POST /api/participants to create a new participant session (instead of /api/teams), etc.  
  Doing this is mostly a search-and-replace effort in routes and client usage. If time permits, it can make the API more semantically correct. Otherwise, document that “team” endpoints now refer to individuals to avoid confusion for future maintainers.

### 4. Front-End UI Modifications

Adapting the client-side is crucial for a smooth user experience in individual mode. We need to adjust text, workflows, and possibly component behavior. The main idea is to remove any notion that the user is working in a team of multiple people, and present it as a personal session. Here’s how we’ll handle each relevant component:

* **TeamModal -> Session Start Modal:** The TeamModal.tsx that handles team creation/joining should be reworked for individuals. Changes include:
* **Text and Labels:** Rename prompts like “Join or Create Team” to “Start or Resume Your Session.” Replace any mention of "Team Code" with "Your Code" or "Session Code", and "Team Name" with either "Your Name" or perhaps "Session Name/Label." Given that for individuals the name could simply be their own name, we might prompt “Enter your name” when starting a session. This will correspond to the name field stored in the database. It can help instructors identify participants (e.g., a list of names instead of just codes). If privacy is a concern, it could be a nickname or even omitted, but likely a name is useful.
* **Creation vs Joining Logic:** In team mode, creating a team involved choosing a team name and code, and joining involved entering an existing team code (and possibly a name to identify yourself within the team, though the code doesn’t indicate that in the backend – there was no separate notion of members). For individual mode:
  + **“Create New Session”:** The user might only need to provide their name. The code could be auto-generated and displayed to them (e.g., “Your session code is ABC123”). Alternatively, we let them choose a code as before. If we allow custom codes, keep the field but emphasize it’s a personal code they’ll use to log back in, not something to share. The form submission will POST to /api/teams as before, then log the user in.
  + **“Resume Session”:** The user just enters their code to resume. This triggers a fetch of their data by calling GET /api/teams/{code}. If found, the UI can proceed to log in. If the code is not found, show an error ("Invalid code"). This is the same flow as join-by-code, but we’ll frame it as resuming your own work rather than joining someone else’s team. We should remove any UI element that suggests multiple people can use the same code simultaneously (even though they technically could, we want to encourage one-code-per-person).
* **Avatar Selection:** In the current system, when creating a team, an avatar is auto-assigned from a set, and users can later change it. We might introduce an option for the user to pick an avatar during creation to personalize from the start. For simplicity, we could also stick to auto-assign on creation (as it does now via selectRandomAvatar[[9]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L76-L84)) and then let them change it on the next screen. The modal could show the assigned avatar (“You have been assigned an avatar, you can change it later”) or give a choice. This is a UX decision – not mandatory to functionality. Regardless, ensure the user understands the avatar is just an icon representing them.
* **NavigationHeader (Top Bar):** This component likely shows the current team’s name and avatar, and possibly a way to switch or logout. For individual use:
* It should display the participant’s name (or "Session: [code]" if we prefer showing code). Ideally, show the name and perhaps the code in parentheses for reference. For example: “👤 Alice (Code: ABC123)”.
* If the original design allowed switching between multiple teams (maybe for a user who might be on more than one team or for an admin to switch context), we can simplify that. A participant in our scenario will typically only have one session active. We can remove any team-switch dropdown. Instead, provide a straightforward **“Logout” or “End Session”** button so they can exit and possibly start a new session or log in with a different code if needed. This will call the /api/auth/team/logout endpoint and then likely bring the user back to the start modal.
* Ensure the header still shows the avatar, and maybe make it clickable to trigger the AvatarSelector (unless that’s accessed elsewhere). For instance, clicking the avatar icon could open the avatar change dialog, which then calls the avatar update API.
* All terminology in tooltips or labels should say “session” or “profile” instead of team. E.g., if there was a tooltip “Switch Team”, change to “Switch Session” or remove it if that feature goes away.
* **AvatarSelector Component:** This can remain largely the same. We will use it as a profile avatar picker for the individual. It should call the same PATCH /api/teams/:id/avatar route to save the selection[[61]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L226-L234)[[17]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L232-L240). We might adjust any heading text (like “Choose a team avatar” -> “Choose your avatar”). Also, ensure that after a user picks an avatar, the UI updates to show it in the NavigationHeader. This likely already works by updating the team state in the app.
* One consideration: if previously avatars were unique per team (the code tries to avoid recent avatar collisions[[66]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L70-L78) by not assigning the same icon to recently created teams), with many individuals, some collisions will happen eventually. This isn't a big issue (it was likely just to give each team a distinct icon at an event or class). We can keep that logic as a nice-to-have uniqueness touch.
* **Phase Pages (Phase.tsx / PhasePage.tsx):** This is where the main content of each section (phase) is displayed and interacted with. We must ensure it works for a single user:
* It should retrieve the participant’s current phase and corresponding data. Likely it calls an API to get phase config (/api/configs/phase-{id}.json) and any saved phaseData for that phase. We should double-check that it references the correct team ID (which in an individual context is their ID). In the team system, it may get teamId from context or from the logged-in session. That will remain the same source – the user’s teamId is stored in the session (and possibly also in the app state after login). So the component can request, for example, GET /api/phase-data/{teamId}/{phaseNumber} to load any existing work[[67]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L254-L263). This doesn’t change, except now teamId refers to the individual’s ID.
* Check for any collaborative features: For instance, was there a notion of multiple team members each doing something on this page? The code as given doesn’t show explicit multi-user handling; it seems each team has a singular state per phase. That’s fine for one person. If any UI text said “Discuss with your team” or similar, those need to be revised to suit a single person (“Reflect on this step” or instructions in singular).
* The progression logic (e.g., moving to the next phase) should continue to use updateTeamPhase. When the individual finishes a phase and clicks "Next", the app likely calls PATCH /api/teams/:id/phase to increment the phase[[15]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L208-L216), and then navigates to the next phase page. This should work identically.
* If sections are truly independent (atomized), we might allow the UI to navigate freely between phases. In that case, we might not update currentPhase sequentially but rather mark phases as completed via markPhaseComplete (there is an endpoint for marking phase complete[[68]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L278-L286), which sets a completion timestamp in phaseData). The current code has markPhaseComplete in storage[[69]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L226-L234)[[70]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L232-L240), but it’s not clear if it’s used in the UI. Possibly it was used to lock in that a team finished a phase. We could use it similarly if needed (for example, to mark section done for tracking even if they can jump around).
* For now, assuming a largely sequential flow, we keep it simple: one phase leads to next. The UI just needs to replace any team references (“Team Progress” -> “Your Progress”, etc.) and ensure the styling still looks good when showing a single name or avatar (likely fine).
* **Results/Showcase Page (Showcase.tsx / Results.tsx):** The code references a Showcase.tsx and Results.tsx in the search results. This suggests that at the end there might be pages to display all teams’ submissions and the voting results. These should be updated to reflect individuals:
* The showcase page likely lists all submitted projects in a cohort (calls GET /api/showcase/:cohortTag which returns teams that submitted URLs[[71]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L499-L507)). We will still use that, but it will list individual names and links. We must make sure the UI labels them properly (e.g., “Participants in Cohort A who submitted their projects” instead of "Teams that submitted"). If it shows team names, those are now participant names.
* The voting page (if used) will allow a logged-in participant to vote for others. The logic and UI can remain, just ensure it doesn’t call participants “teams” in instructions. The server-side already prevents a user from voting for themselves and requires they submit their own project first[[72]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L526-L534)[[51]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L564-L572) – these conditions still make sense (one shouldn’t vote without completing their own work).
* The results page likely shows the outcome of votes (ranking of best projects). This can remain unchanged in logic, but again, label adjustments (e.g., “Top Projects” rather than "Top Teams") should be made.

In summary, the front-end changes are mostly **terminology and UX flow adjustments**. The underlying API calls and state management can remain largely as is: - We keep using the token-based authentication stored in localStorage (as accessToken perhaps). The App.tsx will, on mount, check if the user has a saved token and try to auto-login by hitting /api/auth/team/status. This currently returns the full team if the session is valid[[73]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L144-L152). We will maintain that so users stay logged in across refreshes. If the token is absent or invalid, the app will show the start modal. This logic doesn’t change, except maybe naming (we can call the local storage key something like sessionToken instead of teamToken). - Continue to use the same global state or context to store the logged-in participant’s info (id, name, code, currentPhase, etc.). This was likely stored under a “team” object; we can rename it internally if we want (to “user” or “participant”), but it’s not strictly necessary to function. Just ensure all components use the correct data.

By implementing these changes, the UI will present the application as a personal workspace for each user, while leveraging the robust team-based logic under the hood for saving state and coordinating with the backend.

### 5. Instructor/Admin Interface Enhancements

You indicated you’d like to keep instructor visibility and admin functionality, which is wise – it’s a strength of the platform that instructors can monitor progress and manage the experience. We will adapt the admin interface (Admin.tsx page and related components) to suit individual participants:

* **Terminology and Data Display:** The admin UI should refer to “Participants” or “Users” instead of teams. For example, if it had a heading “Team Management”, change it to “Participant Management”. In lists, show each participant’s name and code. The backend is already providing the data via endpoints like /api/admin/cohorts/:tag/teams\_status which returns an array of team objects[[74]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L483-L491). Each object includes fields like name, code, currentPhase, etc., plus the computed flags for submission/vote status. We should update the front-end to utilize these:
* Show each participant’s name (and perhaps their code in a smaller font for reference).
* Show an icon or label if they have submitted their project (the hasSubmitted boolean)[[75]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L486-L494).
* Show if they have voted (if that’s relevant, hasVoted boolean)[[75]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L486-L494).
* We can also display their current phase number so the instructor knows where they are. If the API is enhanced to include currentPhase in the teamStatus output, use that. If not, the admin could fetch the list of teams (participants) via /api/teams and cross-reference IDs, but it’s easier to just include it in the status call or call /api/teams and join data. We will implement it in the API for convenience.
* **Controls:** The admin page likely has controls for creating or archiving cohorts, toggling submissions/voting, and assigning participants to cohorts:
* These functions remain the same. If there’s an interface to create a new cohort, it will call the storage createCohort (no change) and then the UI can assign participants. The only difference is the UI should list individuals not groups when assigning. If it previously allowed selecting multiple teams to assign to a cohort, it can still do that with individuals (the logic of selecting multiple rows and clicking "Assign to Cohort X" stays).
* If there was functionality to force advance a team’s phase or reset something, that might not exist currently. We might consider adding small admin actions like “reset phase to 1” for a participant if needed (e.g., if someone needs to restart). This could be done via the existing updateTeamPhase API from the admin side. It’s an extra feature outside core needs, but good to note as a possibility.
* **Viewing Participant Work:** As mentioned, a valuable addition would be allowing instructors to review what an individual has done in each phase. Currently, the system doesn’t have a dedicated UI for this, but we can build on what’s there:
* The instructor could click on a participant’s name in the admin list and see a detail view. In that view, the app can call /api/phase-data/{teamId} (which returns all phase data for that team)[[65]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L268-L275). This will give an array of all phases with their saved data (likely JSON content of their answers or outputs). The instructor UI could render this in a readable way (if the data is text, show text; if JSON with specific fields, format it accordingly).
* This feature would greatly enhance the “visibility” aspect for instructors, letting them give feedback or ensure each participant did the exercise. It wasn’t explicitly in the original spec, but it's a logical extension when moving to an individual setting where you might want to individually evaluate each person’s work. It’s up to you if this fits your course workflow.
* If implementing this, ensure that the backend either allows the admin to fetch that data (we might add an admin-specific endpoint, or reuse the existing ones since the admin can always log in as admin and theoretically call the endpoints – but those endpoints aren’t admin-protected, they just need a teamId. To avoid confusion, an admin-only endpoint is cleaner).
* **Performance considerations:** With individuals, the number of entries in admin views might increase (e.g., instead of 10 teams you could have 40 individuals). The UI should handle this gracefully:
* Possibly add a search or filter box on the admin page to find a participant by name or code quickly.
* Ensure that if there’s a table, it can scroll or paginate if many participants are present.
* The data flow (fetching all teams in a cohort) returns a list of possibly dozens of entries, which is fine for typical class sizes. If it became hundreds, we might implement pagination on /api/teams or similar. For now, probably not needed, but keep in mind.
* **Admin Permissions:** The admin login system stays as is. Just make sure all new or repurposed features respect that:
* For example, if we introduce new endpoints for viewing participant data, wrap them with ensureAuthenticatedAdmin so regular users can’t call them[[76]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L30-L38)[[77]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L40-L48).
* The existing admin endpoints already have this middleware where appropriate (we saw it on cohort status route[[20]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L481-L489) and the cleanup utility[[78]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L335-L343)). We will do the same for any additions.

Overall, the instructor experience will remain very similar, just oriented toward individuals. They will log in and see a dashboard of all participants, be able to monitor progress and submissions, and manage grouping. By keeping these features, we ensure that even though each participant is working alone, the instructor can still get a class-wide view and intervene or encourage as needed.

### 6. Summary of Key Changes and Recommendations

To ensure nothing is missed, here’s a checklist of the major adaptations we plan to implement:

* **Backend (General):** Continue using the existing team-based logic for individuals. No major code refactor needed in business logic; just treat each team entry as an individual. Optionally refactor naming (teams -> participants) in code for clarity if time permits. Maintain the session-based authentication using codes and tokens (no user/password for participants).
* **Security Improvements:** While not explicitly asked, we recommend a couple of light security enhancements when moving to individuals:
* Make sure that sensitive operations (saving data, updating profile) require the user’s session. Add ensureAuthenticatedTeam where appropriate so one participant cannot accidentally (or maliciously) affect another’s data just by knowing an ID.
* Encourage stronger or system-generated codes to reduce any chance of code guessing between participants.
* Clearly instruct participants to keep their code private (since now it effectively acts as their login key).
* **User Experience Tweaks:** Remove any multi-user phrasing or steps. Ensure the first thing a user sees is an invitation to start a new session or continue an existing one via code – this should be straightforward and welcoming, as these are professionals using the tool as part of a course. Possibly provide a short note like “No account needed – just remember your code to come back later!” to align expectations.
* **Avatar and Personalization:** Keep the fun elements like avatars to maintain engagement. Possibly even allow users to edit their displayed name if needed (though if the name is their actual name, probably fine as set initially).
* **“Atomized” Sections Consideration:** If the course content is reorganized such that sections can be done non-linearly, consider adjusting the progress model:
* We might allow the UI to navigate sections out of order. In that case, the currentPhase field could be less of a gate and more of a record of last visited or highest completed phase. We could allow accessing any phase via URL or menu, but mark incomplete ones and require eventually that all are completed for a final submission.
* The database could track completion of each phase via phaseData.completedAt timestamp[[70]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L232-L240), which we already store when markPhaseComplete is called. We could use that to know which sections are done.
* This is a more complex change, so if not necessary, keeping the linear flow is fine. The phrase "sections are all atomized" might simply mean each section is independent content (which is fine; the platform doesn’t force interactions between sections except sequential order).
* **Testing:** After these changes, thorough testing is needed:
* Create a new session, go through a few phases, logout, and resume on another browser using the code to ensure data persists.
* Test an admin logging in to see that the participant appears correctly in the cohort with accurate status.
* Test avatar changes, project submission, and (if applicable) voting, to ensure those still work with one-person-per-team.
* Verify that an admin can assign individuals to cohorts and that individuals can only see their own data.
* **Documentation:** Update any README or help text in the app to reflect the new usage (e.g., if there was a guide saying “share this code with your teammates,” change it to “use this code to resume your session”). The codebase itself is well-commented; adding comments where we changed semantics (like “team = individual session in single-user mode”) will help future maintenance.

By following this plan, we retain the **best parts** of the core functionality: - Easy, account-free access with a simple code (carry over team code logic). - Persistent state and cross-device resume (leveraging access tokens and stored phase data). - Instructor oversight (cohort grouping, monitoring progress, viewing submissions). - Structured progress through phases or sections of the material. - Fun personalization (avatars, names) and engagement features like voting.

All of this is achieved with relatively small modifications to the existing code, thanks to the clean separation of concerns. The database and APIs don’t inherently care if a team has 5 people or 1 person – it’s just a record per code. We mainly adjust the front-end to treat that record as a single user context and adjust the wording.

#### Final Recommendations on Instructor Visibility/Admin

You expressed interest in keeping the admin features and possibly improving them. Here are a few focused suggestions for that: - **Keep using the Admin Dashboard:** It’s a valuable tool for you as an instructor to see everyone’s progress at a glance. We’ll adapt it to list individuals, but ensure it remains a central place to toggle any course-wide settings (like closing submissions when the deadline passes – controlled by submissionsOpen in cohort[[79]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L42-L49) – or showing results when ready, controlled by resultsVisible[[79]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L42-L49)). - **Adapt UI to Individuals:** Small UI improvements like highlighting a participant row in red if they haven’t submitted but voting is about to close, etc., could be handy. These can be added using the data we have (hasSubmitted = false and maybe a date check). - **Leverage Data for Interventions:** Because you can see currentPhase (once we include it) for each participant, you can identify who is falling behind. We might even add a feature to send a nudge or message to those participants through the platform (though that’s a larger feature, possibly out-of-scope). At minimum, you can manually reach out if you see someone stuck on an early phase. - **Simplify Admin Cohort Management:** If many individuals enroll, giving each a code manually could be tedious. You might create a bunch of participant entries upfront or rely on them to self-register. If you self-register them, you could pre-generate codes and names (e.g., upload a list of names to auto-create entries). Currently, there’s no bulk import feature, but something to consider for convenience. Alternatively, for a large cohort, instruct participants to self-create but ensure codes are unique. The system will warn on duplicates[[80]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L178-L184), but you might want to suggest a format (like “use your first name or nickname as code” knowing duplicates might occur). - **Future Improvement – Role-based Views:** Now that “teams” are essentially individuals, the platform might in the future support a dual role login – participants (with just code) and instructors (with username/password). It already does, but they operate separately. You might consider adding the ability for an instructor to masquerade as a participant (for demonstration purposes or troubleshooting). This could be done by allowing admin to pick a participant from the list and “enter their view” (setting the session.teamId to that participant). This is advanced and would require some backend support (maybe a special admin endpoint to set session to a teamId, or simply providing the accessToken to the admin to use). It’s not necessary but can be useful if, say, a participant says “I have an issue on phase 2” – you could jump into their session to see exactly what they see. For now, you can always ask for their code and log in as them, which is essentially the same thing since no password on their side. That’s one advantage of the code system – even the instructor could use any code to view that session’s data if needed. Just keep that in mind (and be careful with it to respect privacy).

In conclusion, the transition to individual usage should be smooth. The system’s modular design (separating data, logic, and presentation) means we mostly adjust the presentation and slight parts of the logic, without overhauling the core. Individuals will get a seamless experience saving and resuming work across devices – exactly as teams did, but now tailored to a single user. Instructors will retain full visibility and control, with potential for even finer-grained monitoring. By implementing the plan above, the platform will **“work great” for solo participants** in your generative AI training courses, while still encapsulating the collaborative features (like sharing and voting) that can enrich the learning experience.

Each of these changes can be implemented incrementally and tested, thanks to the clear separation in the code (e.g., update the schema/types, then storage, then routes, then front-end). We’ll proceed with these adaptations, ensuring to keep the best parts of the system intact and improving clarity and usability for the new context.

[[1]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L7-L15) [[2]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L22-L30) [[3]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L39-L47) [[4]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L96-L104) [[5]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L108-L116) [[23]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L7-L14) [[24]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L9-L17) [[25]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L13-L17) [[26]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L11-L19) [[27]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L10-L17) [[28]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L12-L15) [[29]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L32-L37) [[30]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L64-L72) [[31]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L100-L108) [[32]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L104-L112) [[33]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L51-L59) [[34]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L52-L60) [[50]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L56-L62) [[79]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts#L42-L49) schema.ts

<https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/shared/schema.ts>

[[6]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L9-L17) [[7]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L16-L20) [[8]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L211-L219) [[9]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L76-L84) [[10]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L80-L88) [[36]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L93-L101) [[37]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L102-L105) [[40]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L108-L116) [[41]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L219-L224) [[42]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L130-L138) [[43]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L120-L128) [[45]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L144-L152) [[46]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L160-L167) [[47]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L174-L182) [[48]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L314-L322) [[49]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L320-L328) [[62]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L160-L168) [[66]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L70-L78) [[69]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L226-L234) [[70]](https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts#L232-L240) storage.ts

<https://github.com/mithorizon7/FoundBootstrapShortened/blob/f08703a8dfb74dedb99403f3afa682bf78e785b0/server/storage.ts>

[[11]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L103-L111) [[12]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L114-L122) [[13]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L199-L207) [[14]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L187-L195) [[15]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L208-L216) [[16]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L222-L231) [[17]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L232-L240) [[18]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L306-L315) [[19]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L319-L325) [[20]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L481-L489) [[21]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L499-L508) [[22]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L513-L522) [[35]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L176-L184) [[38]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L106-L114) [[39]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L118-L125) [[44]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L314-L322) [[51]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L564-L572) [[52]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L114-L123) [[53]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L135-L141) [[54]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L183-L191) [[55]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L163-L171) [[56]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L161-L169) [[57]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L164-L172) [[58]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L170-L175) [[59]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L52-L59) [[60]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L513-L521) [[61]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L226-L234) [[63]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L62-L71) [[64]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L88-L96) [[65]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L268-L275) [[67]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L254-L263) [[68]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L278-L286) [[71]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L499-L507) [[72]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L526-L534) [[73]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L144-L152) [[74]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L483-L491) [[75]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L486-L494) [[76]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L30-L38) [[77]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L40-L48) [[78]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L335-L343) [[80]](https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts#L178-L184) routes.ts

<https://github.com/mithorizon7/FoundBootstrapSeparated/blob/4453cd4f1cbfa060b9d8150b61e95042f56109d4/server/routes.ts>