Google Summer of Code 2022

Learner Group MVP

Pankaj Prajapati

Section 1: About You

What project are you applying for?

Learner Group MVP.

Why are you interested in working with Oppia, and on your chosen project?

I myself coming from a family where me and my siblings were the 1st generation to get proper education have felt the importance of education in life and the various paths and possibilities it opens up. For some time I also took tuitions for kids in the neighborhood and interacted with poor kids on my school trips. I have always wanted to contribute and help these kids learn which could open up opportunities for them and from the time I joined Oppia, it has been like I am finally able to do some good.

While I very much like the work Oppia does, one also very strong reason for me wanting to contribute to Oppia is the community it has. From the time I joined Oppia I never felt out of the place or uncomfortable at any point of time. The people here have always been helping whenever I had some queries or needed some guidance. Due to these reasons I would still like to continue contributing to Oppia even after GSoC like I have been doing before.

As for the Learner Groups MVP project, I feel it would be a big step towards not only providing learners quality content to learn and interact with but also quality guidance along the way. As the project aims I also share the same sentiments that learning with proper guidance and along with your peers keeps the learner motivated, competitive and enjoyable along the way. Along with that, from a technical point of view as it's a large full stack project and a complete package in itself, I believe it will also help me in my future career.

Prior experience

I have prior experience of working as a software developer for around 2 years which I did along with my college studies till last year where I have worked on end to end development of projects and as it was a startup and we had a small team, I got to experience the whole journey of building projects.

Along with this, I have been working with Oppia for around the last 5-6 months, so I have gotten the chance to learn about the codebase and get a deeper understanding of things during this time.

I am part of the LACE quality team and the Internationalization team. I am also the project lead of the User Checkpointing team at Oppia and have merged around 30+ PRs and have opened 10+ issues. I have gotten the chance to work on various frontend issues, server issues, production fixes, writing backend and frontend tests and fixing e2e tests.

Apart from issues, I have had the chance to work on 2 major projects at Oppia, one that has been completed is the “Hacky Translations” project and the other one is that we are still working on, “User Checkpoints” project. I have also reviewed code for frontend and backend changes that were part of the user checkpoints project being the team lead of the project.

Contributions to Oppia:

- Implementation of Hacky translations for the all classroom related pages
- Add hacky translation support for breadcrumbs on classroom related pages
- Fix part of #14829: Added RTL formatting for remaining files.
- Fix #14714: Modified conditional checks for adding content ids
- Fix #14426: Resolved Multiple Choice Options Shuffling Between Attempts

List of all my contributions to Oppia can be found here.

Project size

Large (~350 hours).

Project timeframe

General Timeline of June 13 - September 12.

Contact info and timezone(s)

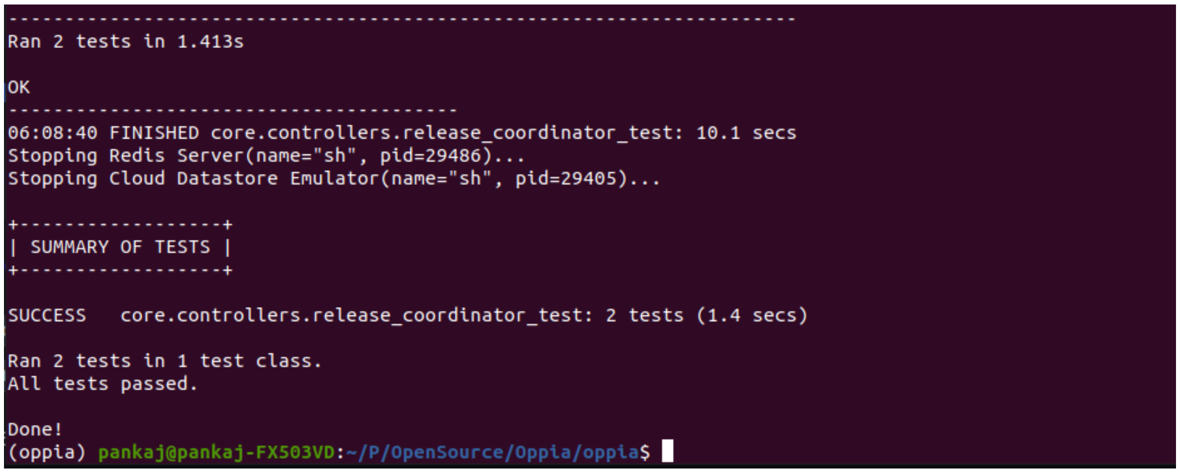
Phone No: (+91) 8510889373
Email: paprajapati9@gmail.com
Prefered mode of communication: Gmail, Gmeet, Google Chat
Timezone: Asia/Kolkata (IST)

Time commitment

I have my summer vacation from 25th May to 19th July, so during this time I would be available for around 40 hr/week, after that as my colleges start I would be available for around 30 hr / week, but I can increase the hours based on the pace of the project.

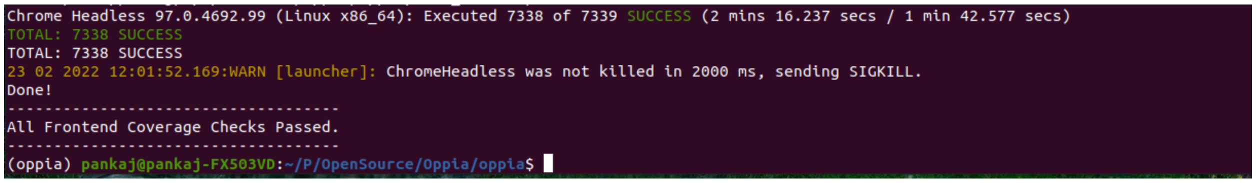
Essential Prerequisites

- I am able to run a single backend test target on my machine.



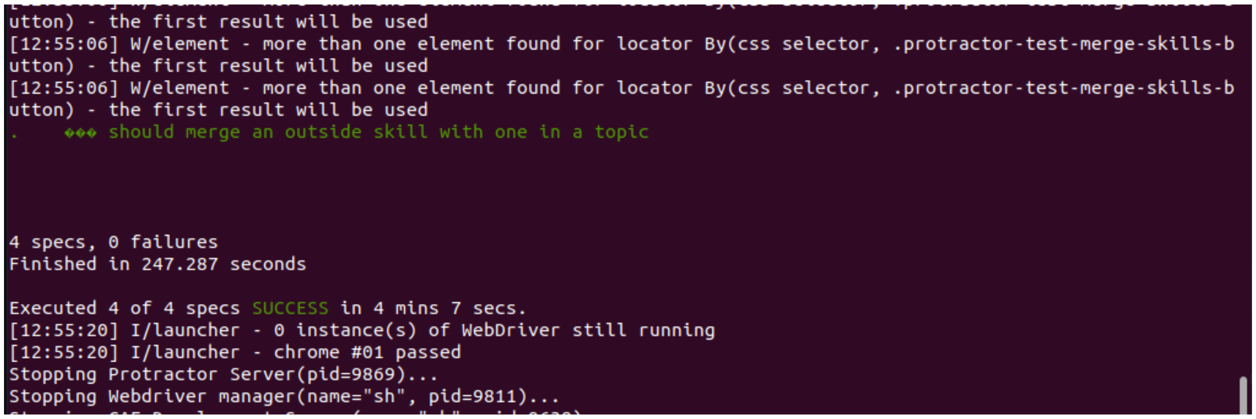
```
----------------------------------------------------------------------
Ran 2 tests in 1.413s

OK
----------------------------------------
06:08:40 FINISHED core.controllers.release_coordinator_test: 10.1 secs
Stopping Redis Server(name="sh", pid=29486)...
Stopping Cloud Datastore Emulator(name="sh", pid=29405)...

+------------------+
| SUMMARY OF TESTS |
+------------------+

SUCCESS   core.controllers.release_coordinator_test: 2 tests (1.4 secs)

Ran 2 tests in 1 test class.
All tests passed.

Done!
(oppia) pankaj@pankaj-FX503VD:~/P/OpenSource/Oppia/oppia$
```

- I am able to run all the frontend tests at once on my machine.



```
Chrome Headless 97.0.4692.99 (Linux x86_64): Executed 7338 of 7339 SUCCESS (2 mins 16.237 secs / 1 min 42.577 secs)
TOTAL: 7338 SUCCESS
TOTAL: 7338 SUCCESS
23 02 2022 12:01:52.169:WARN [launcher]: ChromeHeadless was not killed in 2000 ms, sending SIGKILL.
Done!
-----------------------------------
All Frontend Coverage Checks Passed.
-----------------------------------
(oppia) pankaj@pankaj-FX503VD:~/P/OpenSource/Oppia/oppia$
```

- I am able to run one suite of e2e tests on my machine.



```
utton) - the first result will be used
[12:55:06] W/element - more than one element found for locator By(css selector, .protractor-test-merge-skills-b
utton) - the first result will be used
[12:55:06] W/element - more than one element found for locator By(css selector, .protractor-test-merge-skills-b
utton) - the first result will be used
.    ◆◆◆ should merge an outside skill with one in a topic



4 specs, 0 failures
Finished in 247.287 seconds

Executed 4 of 4 specs SUCCESS in 4 mins 7 secs.
[12:55:20] I/launcher - 0 instance(s) of WebDriver still running
[12:55:20] I/launcher - chrome #01 passed
Stopping Protractor Server(pid=9869)...
Stopping Webdriver manager(name="sh", pid=9811)...
```

Other summer obligations

I do not have any such obligations this summer, plus I am only applying from Oppia in GSoC so things should be fine.

Communication channels

Meeting times : 2 times per week(flexible) [Google-Meet, Discord, or any other platform]
I will try to provide daily updates as much as possible to ensure smooth working and doubts clarification.
[I check my email regularly and can be contacted on any of the above-mentioned platforms]

Section 2: Proposal Details

Problem Statement

Link to PRD (or N/A if there isn't one)	Learner Group v0 PRD
Target Audience	Teachers, Learners, Tutors, Parents
Core User Need	Teachers, tutors, and parents are often highly invested in monitoring the progress and goals of multiple learners at a time, and, as a learning platform, we want to ensure our platform can enable these stakeholders to guide their learners if they are available and willing to do so. While Oppia believes that ultimately learning should be motivated and driven by the student's own goals and curiosity, we also believe that learners often are more motivated and engaged when they do so with the guidance and support of said stakeholders.
What goals do we want the solution to achieve?	• Facilitator can provide recommendations and goals to help guide learners and monitor their progress. • Learners can learn by completing goals assigned and can optionally share their progress so that facilitators can be able to provide additional support as needed.

Section 2.1: WHAT

Key User Stories and Tasks

#	Title	User Story Description (role, goal, motivation)	Priority[1]	List of tasks needed to achieve the goal (this is the "User Journey")	Links to mocks / prototypes, and/or PRD sections that spec out additional requirements.
1	Joining a Group via direct invite	**As a** Learner, **I need** to be able to join a learner group via direct invite on my learner dashboard	Must Have	Facilitator will send an invite to the learner to join a group through their username.	Add new member button Add new member modal
				The learner will receive the invite on their learner group page which they can accept/reject	Invitations view Decline Invitation
				The invite will have an option to view	Invitations view

[1] Use the **M**o**SC**ow system ("Must have", "Should have", "Could have"). You can read more here.

				more details about the learner group, clicking which will open an invitation modal for the learner group.	
				The invitation modal will show learner group name, description, and group facilitator's display name, it will also have the option to accept/reject the group invitation Edge Case Scenario: 1. **Learner deletes their account after they get the invite but before they accept it :** Remove their user_id from the invitations of the related learner groups. 2. **Learner deletes their account after they get the invite after they accept it :** Remove their user_id from the members of the related learner groups. 3. **Learner group gets deleted before the learner accepts the invitation :** Remove the learner group id from the user's invitations.	Modal view
2	Setting Progress Sharing Permission s	**As a** Learner, **I need** to be able to choose and change my progress sharing permissions	Must Have	After accepting a group joining invitation, learners will be asked to set whether they would like to share their learning progress with their learner group facilitator.	Privacy settings modal (expanded) Privacy settings modal
				The preference of sharing progress permissions can be changed at any point of time by learner from learner group preference settings	Privacy Settings
				Facilitators being able to see a learner's progress will depend on what permissions the learner has set.	Learner group page (facilitator view)
3	Seeing Progress in Learner Group Page	**As a** Learner, **I need** to be able to see which assigned stories I've completed and the level of mastery I have of assigned skills.	Must Have	On the group page there are 3 tabs; home, assigned syllabus, skill proficiency which show related details to you. Select any of these tabs and see your progress. Home tab is the default tab. Edge Case Scenario: 1. **A syllabus item present in the**	Learner's Progress

				learner group was deleted/unpublished from oppia : In every get request for the learner group view for either learner or facilitator we check if all the syllabus items exist or not, if any of them do not exist, we remove them from the learner group syllabus. **2. Learner tries to see progress of a deleted group though entering the url of a deleted group :** Show Page not found error.	
4	Start Practice Sessions from Learner Group Page	**As a** Learner, **I need** to be able to start or continue skills that have been assigned by my learner group facilitator.	Must Have	From the skill proficiency tab, after seeing the progress, the learner can choose to continue practicing a skill or choose a new skill to work upon.	Start Practicing Button
				After clicking on the skill card, learner will be redirected to the practice tab of it's topic with the skill being preselected Edge Case Scenario: 1. **A syllabus item present in the learner group was deleted/unpublished from oppia or the facilitator removed it from the group :** We will show the learner the updated syllabus items(currently part of the syllabus and are present in oppia) to choose from.	
5	Start Lessons from Learner Group page	**As a** Learner, **I need** to be able to directly start or continue lessons (which are assigned by story, not individual lessons) that have been assigned by my learning group facilitator.	Must Have	From the assigned syllabus tab, after seeing the progress, the learner can choose to continue learning a lesson or choose a new lesson to work upon.	Assigned syllabus
				After clicking on the lesson card, learner will be redirected to the exploration page of that lesson in the story view Edge Case Scenario: 1. **A syllabus item present in the learner group was deleted/unpublished from oppia or the facilitator removed it from the group :** We will show the learner the updated syllabus items(currently part of the syllabus and are present in oppia) to choose from.	

6	Learner View of Learner Group Preferences Page	**As a** Learner, **I should** be able to access and update the Learner Group Preferences	Must Have	From the learner group page open the group preferences page	
				On this page, learner should be able to : 1. See the learner group's information (name, description, and facilitator profile) 2. Change progress sharing preference 3. Leave the learner group 4. See FAQs for Learner Group functionality	Group preferences page (group details) Group preferences page (privacy settings) Group preferences page (faq)
				If the learner chooses to leave a learner group, they should receive a confirmation modal simply confirming that they do intend to leave the group.	Leave Group button
7	Learner Group Creation	**As a** Facilitator, **I need** to be able to create a new learner group.	Must Have	At the teacher dashboard page, the Facilitator will see a button to create a new learner group. When clicking this button a new Create new group page would open.	Create new group button
				When creating, a new group must have a name, short description, and at least one syllabus item (e.g. a skill or story that they're recommending students complete) Edge Case Scenario: 1. **Facilitator enters a wrong username when inviting a user:** Show them an error message saying "Sorry, the username entered is invalid, please try again." **2. Facilitator enters a username when already a member or has been invited before for inviting a user:** Show them an error message saying "Sorry, the user is already a member, please enter a different username." or "Sorry, the user has already been invited, please enter a different username.". **3. Facilitator enters their own username for becoming a member:** Show them an error message saying "Sorry, you are the facilitator of the group and cannot become a member	Create new group page (group details) Create new group page (add syllabus) Create new group (add new user)

				of it. Please enter a different username.” **4. Facilitator tries to create a learner group without adding any syllabus items:** The create group button is unclickable until at least 1 syllabus item is added to the group.	
				After the above step, the facilitator will be able to see the create group button, clicking this will create a group and redirect the facilitator to the created group page.	Group created modal
8	Facilitator view of Teacher Dashboard Page	**As a** Facilitator, **I need** to be able to access the teacher dashboard page	Must Have	In user profile dropdown, along with learner and contributor dashboard links, facilitators can also view teacher dashboard page link	
				Opening this link will open the teacher dashboard page, which contains a list of all learner groups owned by the facilitator with option to create, view and delete the learner groups.	Teacher dashboard page
				The facilitator can set this page as the default page opened after login, this can be done via user preferences page.	
9	Adding Learners to a Group via username	**As a** Facilitator, **I need** to be able to invite learners from their usernames	Must Have	From group preferences page open group members tab	Add new member button
				See a list of all members and a button to add new members. Clicking on this will open a modal with the option to add new members by their usernames.	Add new member modal
				While creating the group, in the 3rd step, the facilitator can invite new members by their usernames. Edge Case Scenario: **Same as the ones mentioned in points 1-3 in Learner Group Creation flow**	Create new group (add new user)
10	Facilitator View of Learner Group Page	**As a** Facilitator, **I need** to be able to see an overall view of my group, all options	Must Have	By clicking the learner group card on teacher dashboard page, facilitator will be redirected to the individual learner group page	Teacher dashboard page

		available and the progress of learners in my group		On this page the facilitator should be able to see the following things: 1. A highlight at the top of the skills and questions that learners in their group are having the most trouble with and that could use additional intervention 2. An option to add or edit their group's syllabus 3. An options to look more deeply at the learning progress of each individual learner - if the learner has granted sharing their progress with their facilitator 4. High level breakdown of the broader group's progress through the assigned syllabus so far (e.g. which lesson they're completing or which lesson they have to complete next) - if the learner has granted sharing their progress with their facilitator Edge case scenarios: **1. Facilitator tries to open a deleted group though entering the url of a deleted group :** Show Page not found error. **2. A syllabus item present in the learner group was deleted/unpublished from oppia :** In every get request for the learner group view for either learner or facilitator we check if all the syllabus items exist or not, if any of them do not exist, we remove them from the learner group syllabus. **3. Facilitator tries to access a learner specific page of a learner who has either turned off their progress sharing permissions or left the group by entering the url :** Show Page not found error.	Learner group page (overview) Learner group page (syllabus) Learner group page (learner specific progress page) Learner group page (learner specific progress tab)
11	Editing the Group Syllabus	**As a** Facilitator, **I need** to be able to add new lessons, skills to the group syllabus and remove existing ones from the group syllabus.	Must Have	From the learner group page, the facilitator can switch to the syllabus tab.	Learner group page (syllabus)
				In this tab, the facilitator will be able to add new syllabus items from searching and adding the items from search bar and filters.	Add new syllabus

				The facilitator can choose at any time to remove a syllabus item. When removing a syllabus item a confirmation modal would pop up to confirm if they really want to remove it. Edge Case Scenario: **1. Facilitator tries to delete all syllabus items from the group post creation:** We allow the deletion but we do not allow saving the update by making the done button of the add new syllabus items page unclickable and add a tooltip saying something like "Learner group must have at least one syllabus item". 2. **All syllabus items in the learner group are removed from oppia:** Show no syllabus items present to the learner's home page and on the learner specific progress page.	Remove syllabus item button Remove syllabus item confirmation popup
12	Facilitator View of Learner Group Preferences Page	**As a** Facilitator, **I need** to be able to update or delete a learner group.	Must Have	From the learner group page, the facilitator can go to the group preferences page by clicking the group preferences button.	Group preference button
				When the facilitator views the Learner Group Preferences page, they'll see the following options: 1. Change the group name 2. Change the group description 3. Invite additional learners to the group 4. Remove learners from the group 5. Delete the group Edge case scenarios: **1. Facilitator deletes a learner group :** Delete the LearnerGroupDataModel corresponding to the learner group and remove the group ID from all LearnerGroupUserData models. All others present in "Adding Learners to a Group via username" flow.	Group preference page Change group details Group members Invite members Revoke invitations Remove members
				When a facilitator chooses the option to delete the group, a confirmation modal should appear	Delete group popup

				confirming that the facilitator intentionally set out to delete the group.	
13	Learner-specific Info Page	**As a** Facilitator, **I need** to be able to see the descriptive progress of each individual learner in my group if the learner has given the access sharing permissions.	Must Have	By clicking the on the learner name on the learner group page, the facilitator will be redirected on the learner-specific info page	View details button
				If NOT granted progress sharing permissions to that group, then the facilitator will only be able to see the learner's basic information, like their profile picture, bio, and user name.	Access not granted
				If granted progress sharing permissions, then the facilitator will be able to see the above and the following (in order of priority) 1. A highlighted set of skills or questions where the learner had the most trouble learning / understanding (based on low ratio of correctly answered to incorrectly answered or skipped questions) ● If available, include the list of misconceptions the student is facing based on their answers to practice or lesson questions ● Include up to 5 most recent questions associated with each skill that were answered incorrectly or skipped 2. The learner's progress through each of the assigned skills (e.g. not started, basic, intermediate, mastered) and lessons (e.g. not started, in progress, completed) ● Facilitator should be able to search for specific skills by name or filter skills based on topic and/or learner mastery Edge case scenarios: 1. **No learners are present in the group :** Show message saying "There are currently no learners present in the group. Please add learners to view their progress.". 2. **All syllabus items in the learner**	Learner specific page Learner specific page details (skill analysis) Learner specific page details (progress in stories)

				group are removed from oppia: Show message saying “No syllabus items present to the learner’s progress. Please add new syllabus items”. 3. **Some syllabus items get deleted:** Show progress of users in the syllabus items currently present.	

Technical Requirements

Additions/Changes to Web Server Endpoint Contracts

#	Endpoint URL	Request type (GET, POST, etc.)	New / Existing	Description of the request/response contract (and, if applicable, how it’s different from the previous one)
1.	/createlearnergrouphandler/data	POST	New	Creating a new learner group. When a new group is created it will send a POST request to this endpoint and a new group is created in the **LearnerGroupDataModel**. This post request will contain the following information: 1. title 2. description 3. facilitator (username) 4. invitations(List of usernames) 5. syllabus
2.	/updatelearnergrouphandler/data	PUT	New	Updating an existing learner group. When group preferences are updated by facilitator or group syllabus is changed or new learner is added or existing is removed. A POST request is sent to this endpoint which updates the data in **LearnerGroupDataModel.** This post request will contain the following information: 1. groupId 2. title 3. description 4. facilitator (username) 5. members(List of usernames) 6. invitations(List of usernames) 7. syllabus
3.	/deletelearnergroup	DELETE	New	Deleting an existing learner group

	andler/<groupId>			This post request will contain the following information: 1. groupId
4.	/joinlearnergrouphan dler/data	PUT	New	User accepting a learner group invitation This post request will contain the following information: 1. groupId 2. learner_username 3. invitation_accepted (True/False)
5.	/declinelearnergroupi nvitationhandler/data	PUT	New	User declining a learner group invitation This post request will contain the following information: 1. groupId 2. learner_username 3. invitation_accepted (True/False)
5.	/exitlearnergrouphan dler/data	PUT	New	User exiting a learner group This post request will contain the following information: 1. groupId 2. learner_username
6.	/userinfohandler/data	PUT	Existing	Learner updates progress sharing permissions for a group. The POST request contains the following parameters: 1. progress_sharing_permissions (A map of groupId and true/false indicating the progress sharing permissions are true/false)
7.	/learnerviewoflearner grouphandler/<groupI d>	GET	New	Get all info regarding a group. The following parameters will be returned : 1. groupId 2. title 3. description 4. facilitator (username) 5. members(List of usernames) 6. syllabus
8.	/facilitatorviewoflearn ergrouphandler/<grou pId>	GET	New	Get all info regarding a group. The following parameters will be returned : 1. groupId 2. title 3. description 4. facilitator (username) 5. members(List of usernames) 6. invitations(List of usernames) 7. syllabus
8.	/learnergroupsyllabus	GET	New	Get all syllabus items that are to be shown to the

	handler/data			facilitator while selecting the group syllabus. The request will pass groupId and filterArgs as parameters for filtering the syllabus items to be shown. The filterArgs parameter will have the following keys: 1. keyword 2. type 3. category 4. language The following parameters will be returned in the GET request: 1. stories_summary_dicts 2. skills_summary_dicts
9.	/learnergroupsdashboardhandler	GET	New	Get all learner groups related info of a learner.. It accesses **LearnerGroupUserModel** to get this data. The GET request returns the following parameters: 1. invitationLearnerGroupSummaries (list of learner group summaries that learner is invited to join) 2. learnerGroupSummaries (list of learner group summaries that learner is part of) 3. progressSharingPermissions (A map of groupId and true/false indicating the progress sharing permissions are true/false)
10	/teacherdashboardhandler	GET	New	Get all learner groups related info of a facilitator.. It accesses **LearnerGroupUserModel** to get this data. The GET request returns the following parameters: 1. learnerGroupSummaries(list of learner group summaries that facilitator is owner of)

Calls to Web Server Endpoints

#	Endpoint URL	Request type (GET, POST, etc.)	Description of why the new call is needed, or why the changes to an existing call is needed
1.	/preferenceshandler/data	PUT	Update default_dashboard user settings for the Teacher Dashboard Page
2.	/preferenceshandler	GET	Get default_dashboard user settings for the Teacher Dashboard Page to set it as the home page

UI Screens/Components

#	ID	Description of new UI component	i18n required?	Mock/spec links	A11y requirements
1.	/teacher-dashboard	A page showing all learner groups owned by the facilitator with an	Yes	Teacher dashboard page	

		option to create, view and delete learner groups.			
2.	/learner-groups	A page showing all learner groups the learner is part of and all the invitations the learner has.	Yes	Learner group page	
3.	/learner-groups/<groupId>	Learner group specific page for members of a learner group.	Yes	Group Specific Page (learner view)	
4.	/create/learner-groups/<groupId>	Learner group specific page for the facilitator of a learner group.	Yes	Group Specific Page (facilitator view)	
5.	/learner-groups/preferences/<groupId>	Learner view of the learner group preferences page for a specific group.	Yes	Group Preferences Page (learner view)	
6.	/create/learner-groups/preferences/<groupId>/p	Facilitator view of the learner group preferences page for a specific group.	Yes	Group Preferences Page (facilitator view)	

Data Handling and Privacy

#	Type of data	Description	Why do we need to store this data?	Anonymized?	Can the user opt out?	Wipeout policy	Takeout policy
1.	User progress data	We are exposing other learners' data to the facilitator, who can see the learner's progress if they have opted to share it.	We are not really storing anything new, we are just letting the facilitator view what was already stored	No	Yes, the learner can at any point of time turn off or on progress sharing settings from group preferences	Since we are not storing user's progress in learner groups separately, it is not applicable for any learner groups related fields.	Since we are not storing user's progress in learner groups separately, it is not applicable for any learner groups related fields.
2.	Learner groups a learner belongs to	We are storing all groups that a learner is part of.	We need this to enable access of the learner group for the learner	No	Yes, as soon as the learner leaves the group or is removed from the group, their data regarding that learner group is removed.	When the learner gets deleted, delete their corresponding LearnerGroupUserData Model and remove their user id from the members field of all LearnerGroupData models	LearnerGroupUserData model contains data directly corresponding to learner groups the user is part of hence this model will be exported. LearnerGroupData Model also contains data corresponding to the user but is a ONE_INSTANCE_SHARED_ACROSS_USERS, so all keys except for facilitators, members and invitations are exported and the "members" field is exported as "member" with just the current user's user id.

3.	Learner groups a learner is invited to join	We are storing all groups that a learner has been invited to join.	We need this to allow users to join a learner group only if they have been invited to join it.	No	Yes the user can decline invitations and the corresponding data stored would be removed	When the learner gets deleted, delete their corresponding LearnerGroupUser Data Model and remove their user id from the invitations field of all LearnerGroupData models	LearnerGroupUserData model contains data directly corresponding to learner groups the user is part of hence this model will be exported. LearnerGroupData Model also contains data corresponding to the user but is a ONE_INSTANCE_SHARED_ACROSS_U SERS, so all keys except for facilitators, members and invitations are exported and the "invitations" field is exported as "invitation" with just the current user's user id.
4.	Learner groups a facilitator has created	We are storing all groups that a facilitator has created.	We need this to allow facilitators to actually be able to access the learner groups they have created as a facilitator and not as a learner.	No	Yes the user can delete their learner groups and the corresponding data stored would be removed.	When the facilitator gets deleted, delete LearnerGroupData of all groups owned by the facilitator and remove their group_ids from invitations and members field of all LearnerGroupUser Data models.	LearnerGroupUserData model contains data directly corresponding to learner groups the user is part of hence this model will be exported. LearnerGroupData Model also contains data corresponding to the user but is a ONE_INSTANCE_SHARED_ACROSS_U SERS, so all keys except for facilitators, members and invitations are exported and the "facilitators" field is exported as "facilitator" with just the current user's user id..

Other Requirements

We should have a flag variable to turn on/off learner group feature, for this we will have a config property

```
LEARNER_GROUPS_ARE_ENABLED = ConfigProperty(
        'learner_groups_are_enabled', BOOL_SCHEMA,
         'Enable learner groups feature", False)
```

This will be inside the config_domain.py file and if this is false the frontend would show a page not found error for all learner group related pages and would work normally if the value is true.

Section 2.2: HOW

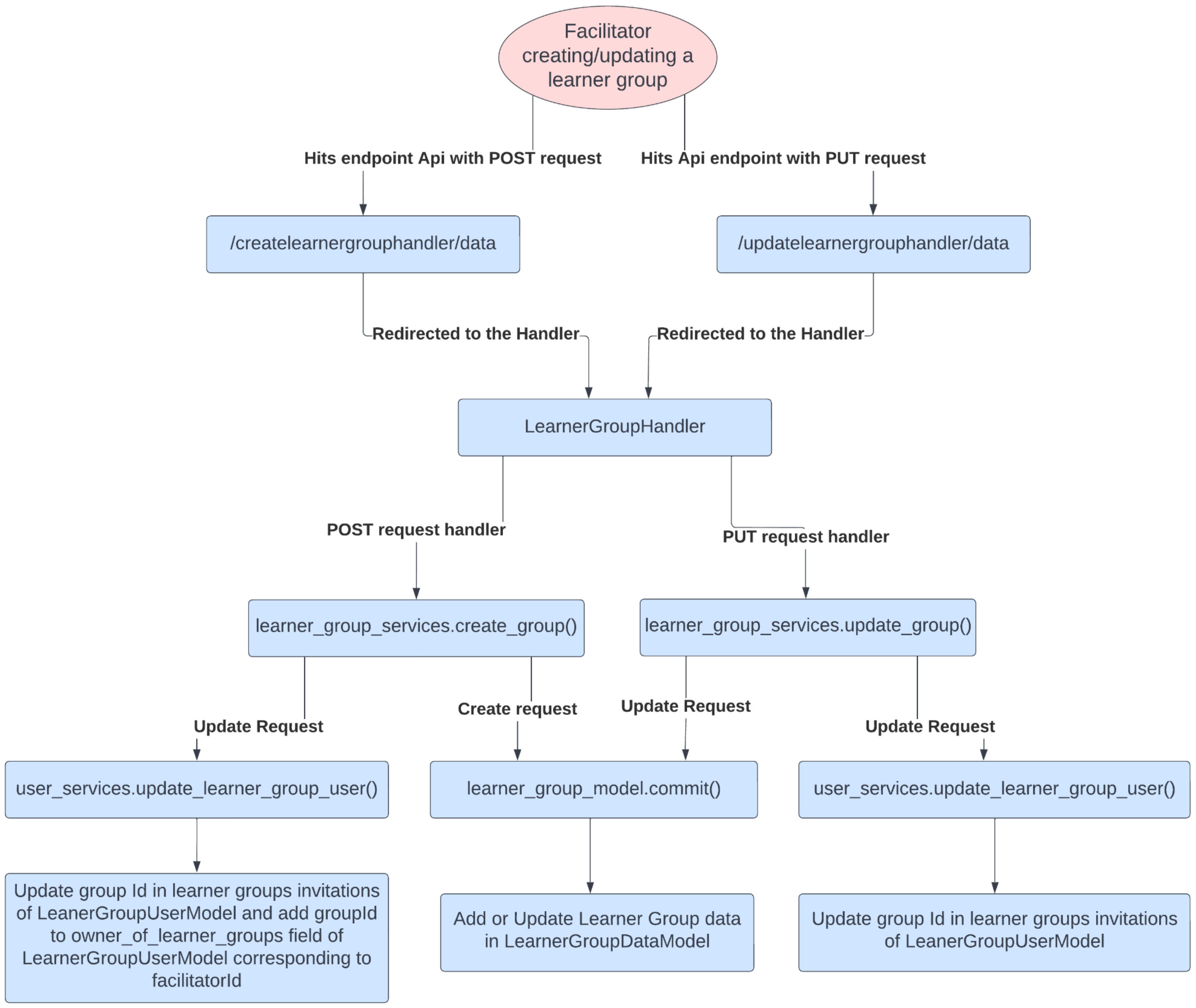
Existing Status Quo

While we do track the progress of our learners, currently we do not have any concept of group learning and directional learning where a learner is suggested to learn specific assigned stories and skills to learn about a specific concept.

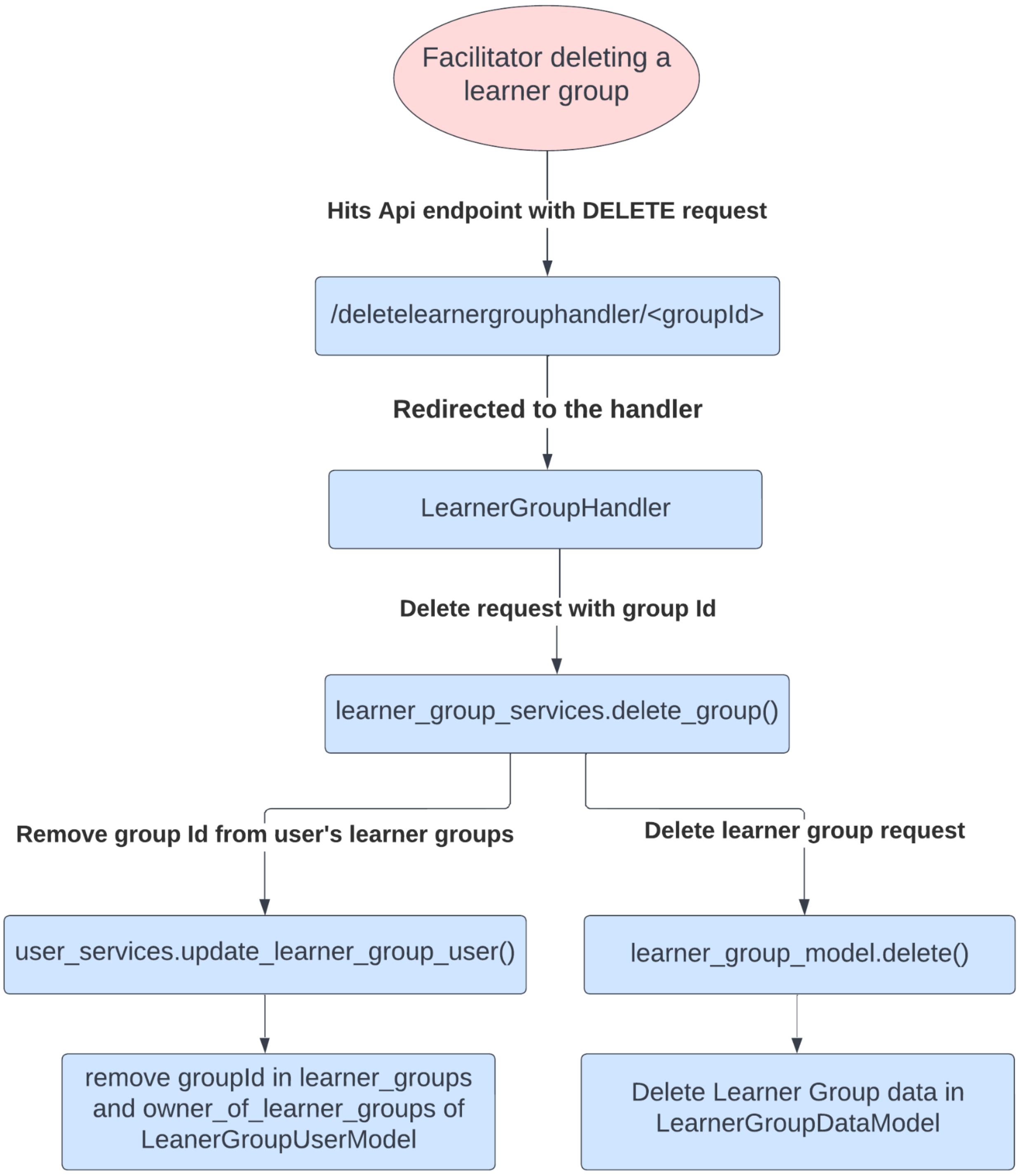
Solution Overview

Learner group CRUD operations:

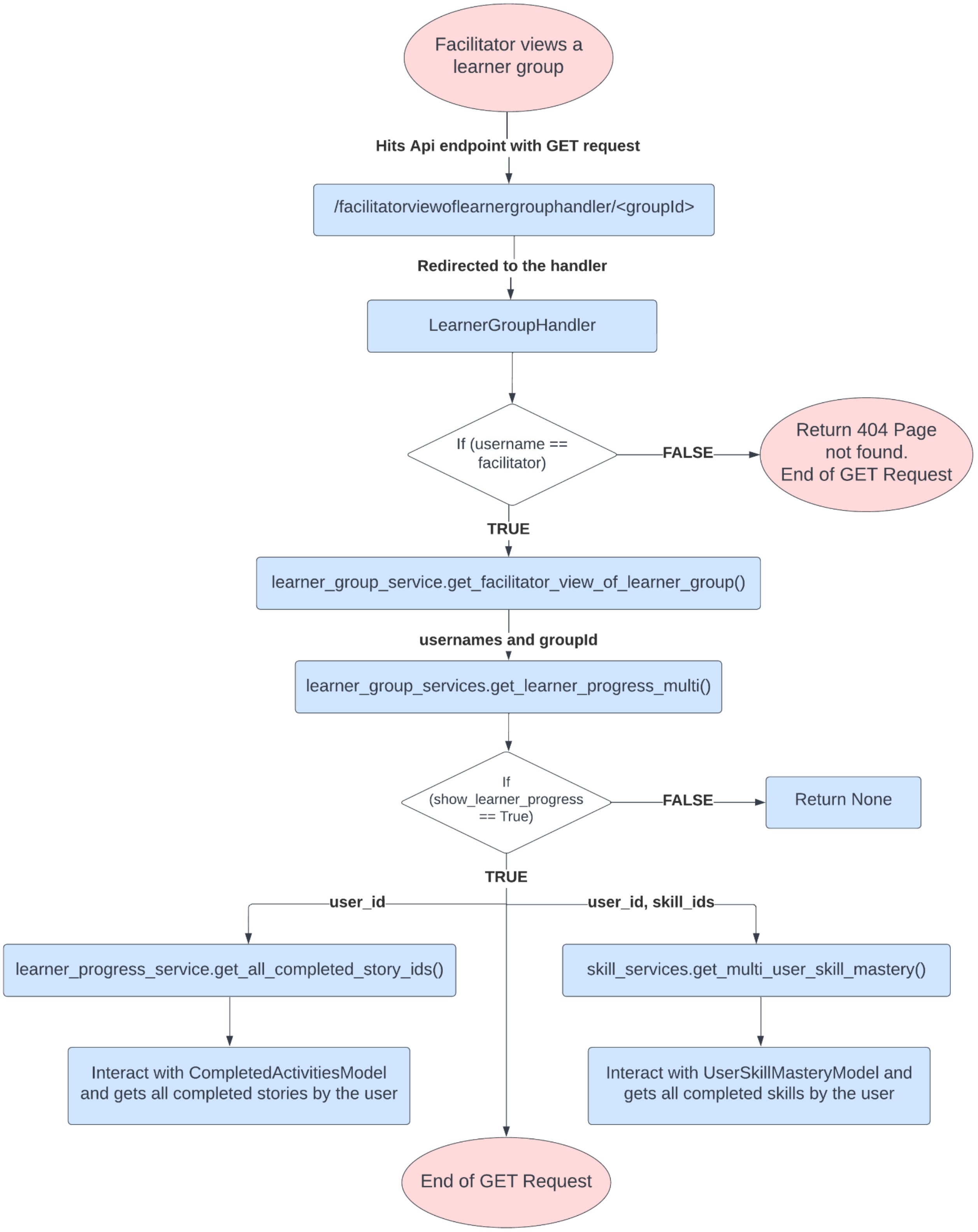
- POST Request workflow when the facilitator creates or updates a learner group:


Facilitator creating/updating a learner group
Hits endpoint Api with POST request
Hits Api endpoint with PUT request
/createlearnergrouphandler/data
/updatelearnergrouphandler/data
Redirected to the Handler
Redirected to the Handler
LearnerGroupHandler
POST request handler
PUT request handler
learner_group_services.create_group()
learner_group_services.update_group()
Update Request
Create request
Update Request
Update Request
user_services.update_learner_group_user()
learner_group_model.commit()
user_services.update_learner_group_user()
Update group Id in learner groups invitations of LeanerGroupUserModel and add groupId to owner_of_learner_groups field of LearnerGroupUserModel corresponding to facilitatorId
Add or Update Learner Group data in LearnerGroupDataModel
Update group Id in learner groups invitations of LeanerGroupUserModel

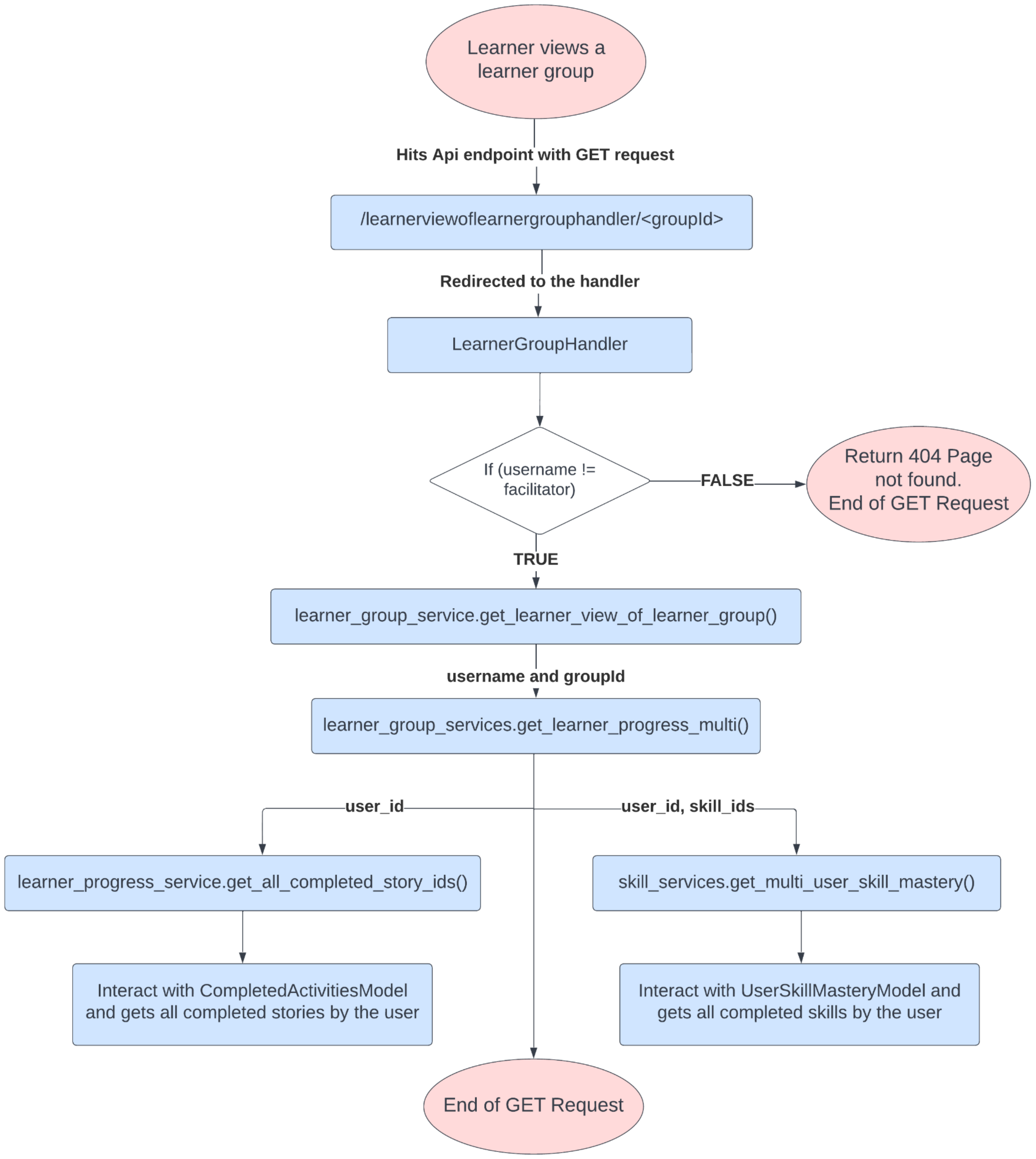
- DELETE Request workflow when the facilitator deletes a learner group:


Facilitator deleting a learner group
Hits Api endpoint with DELETE request
/deletelearnergrouphandler/<groupId>
Redirected to the handler
LearnerGroupHandler
Delete request with group Id
learner_group_services.delete_group()
Remove group Id from user's learner groups
Delete learner group request
user_services.update_learner_group_user()
learner_group_model.delete()
remove groupId in learner_groups and owner_of_learner_groups of LeanerGroupUserModel
Delete Learner Group data in LearnerGroupDataModel

- GET Request workflow when the facilitator views a learner group:

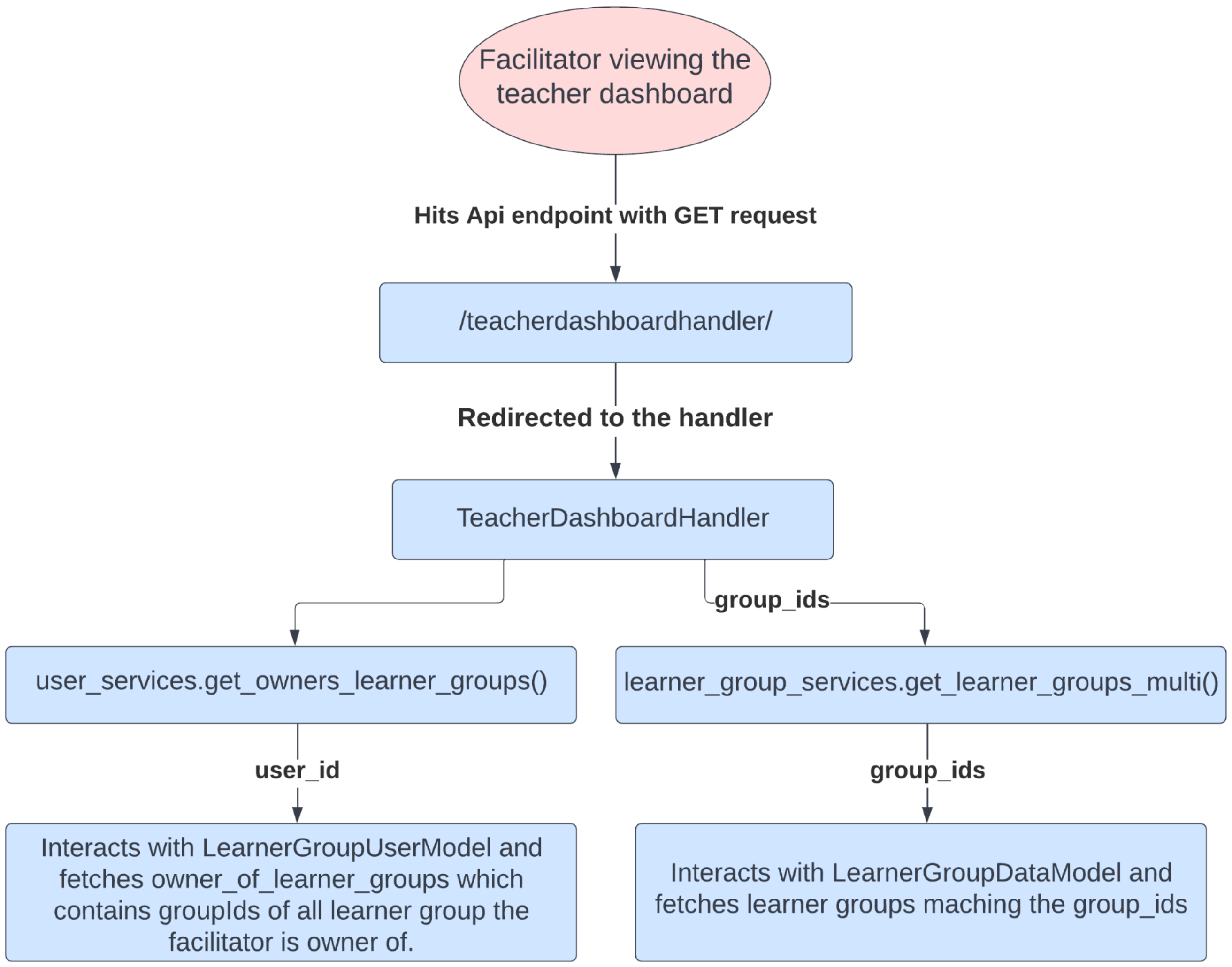

Facilitator views a learner group
Hits Api endpoint with GET request
/facilitatorviewoflearnergrouphandler/<groupId>
Redirected to the handler
LearnerGroupHandler
If (username == facilitator)
FALSE
Return 404 Page not found. End of GET Request
TRUE
learner_group_service.get_facilitator_view_of_learner_group()
usernames and groupId
learner_group_services.get_learner_progress_multi()
If (show_learner_progress == True)
FALSE
Return None
TRUE
user_id
user_id, skill_ids
learner_progress_service.get_all_completed_story_ids()
skill_services.get_multi_user_skill_mastery()
Interact with CompletedActivitiesModel and gets all completed stories by the user
Interact with UserSkillMasteryModel and gets all completed skills by the user
End of GET Request

- GET Request workflow when the learner views a learner group:


Learner views a learner group
Hits Api endpoint with GET request
/learnerviewoflearnergrouphandler/<groupId>
Redirected to the handler
LearnerGroupHandler
If (username != facilitator)
FALSE
Return 404 Page not found. End of GET Request
TRUE
learner_group_service.get_learner_view_of_learner_group()
username and groupId
learner_group_services.get_learner_progress_multi()
user_id
user_id, skill_ids
learner_progress_service.get_all_completed_story_ids()
skill_services.get_multi_user_skill_mastery()
Interact with CompletedActivitiesModel and gets all completed stories by the user
Interact with UserSkillMasteryModel and gets all completed skills by the user
End of GET Request

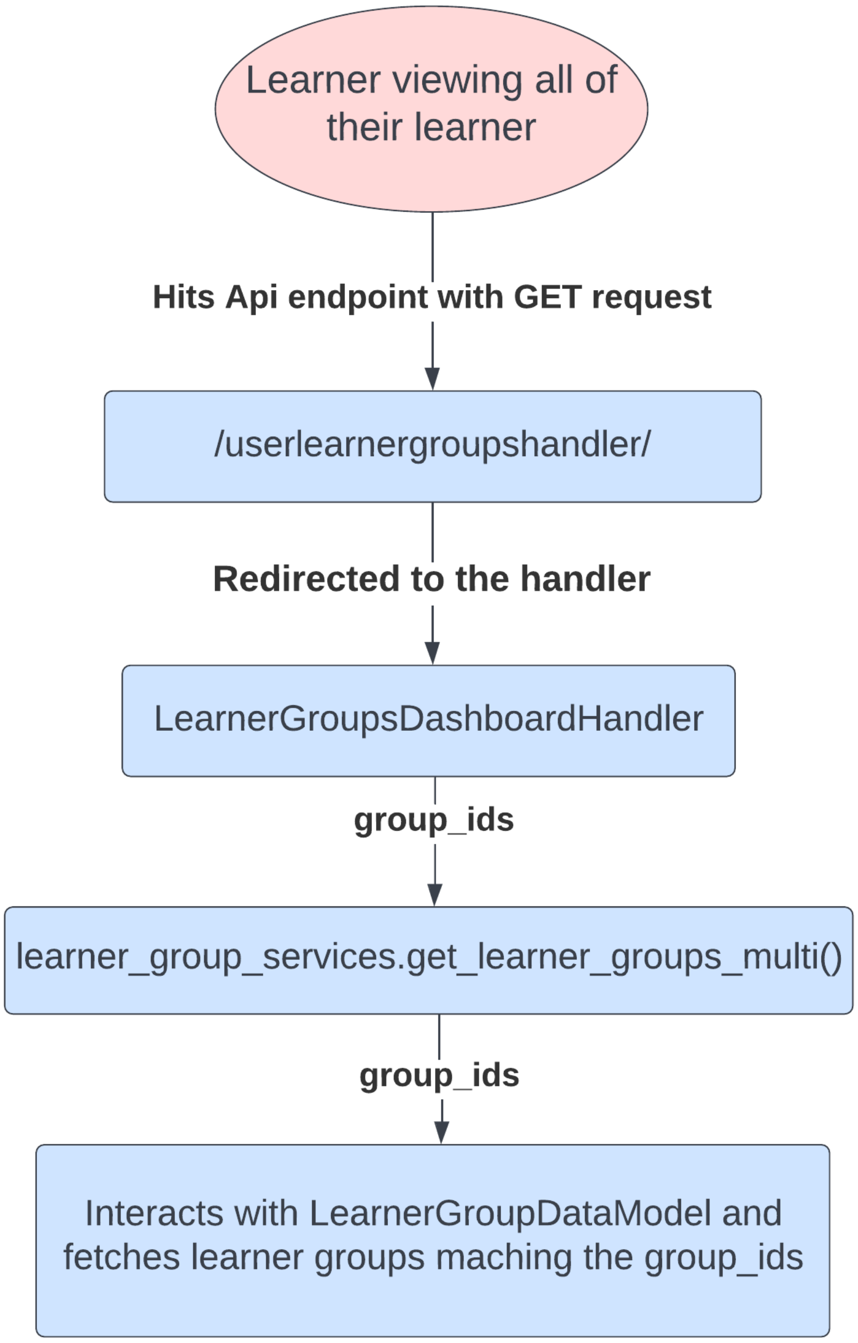
View Teacher Dashboard:

- GET Request workflow when the facilitator views the teacher dashboard:


Facilitator viewing the teacher dashboard
Hits Api endpoint with GET request
/teacherdashboardhandler/
Redirected to the handler
TeacherDashboardHandler
group_ids
user_services.get_owners_learner_groups()
learner_group_services.get_learner_groups_multi()
user_id
group_ids
Interacts with LearnerGroupUserModel and fetches owner_of_learner_groups which contains groupIds of all learner group the facilitator is owner of.
Interacts with LearnerGroupDataModel and fetches learner groups maching the group_ids

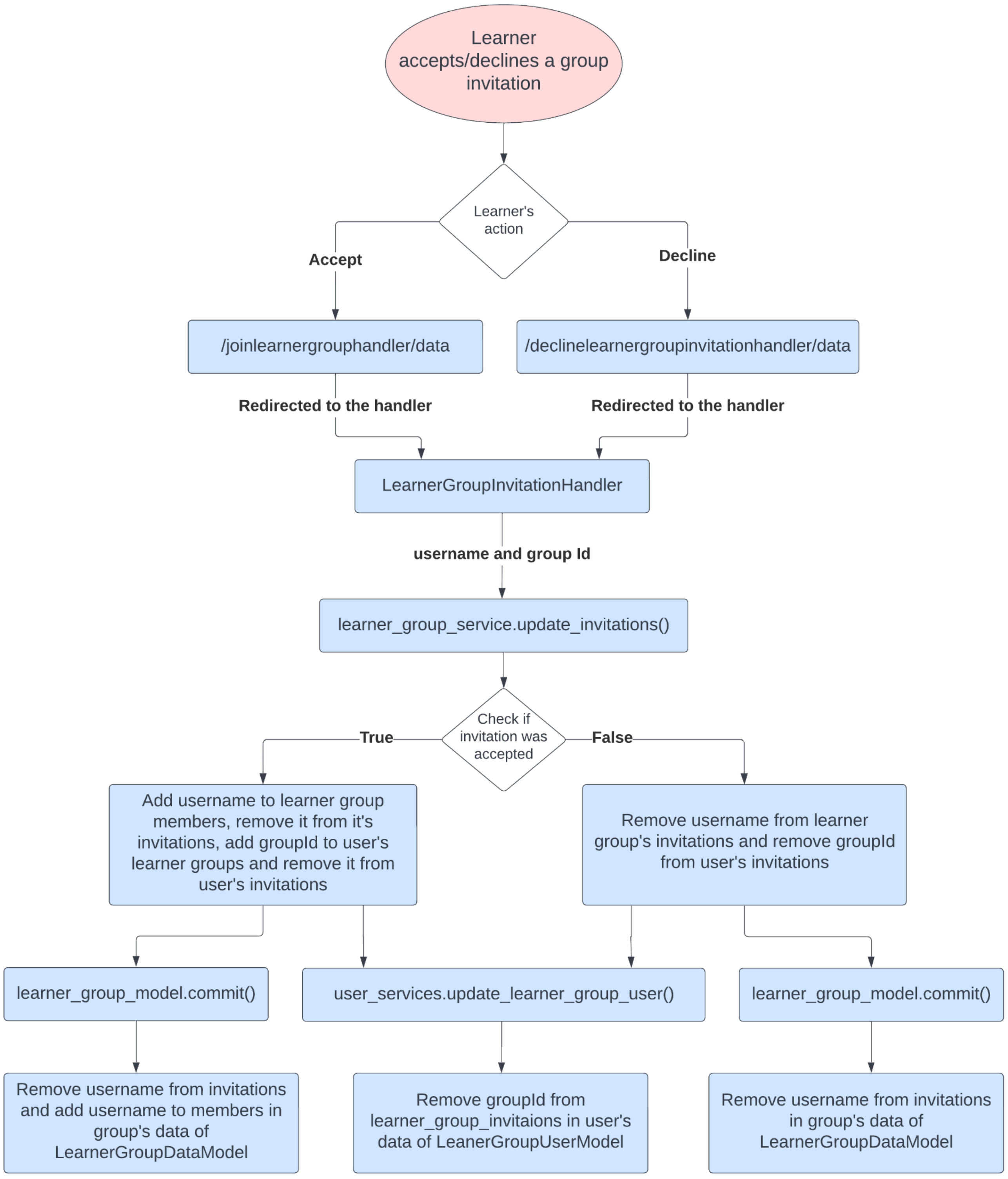
View all learner groups of the learner:

- GET Request workflow when the learner views the learner groups page:


Learner viewing all of their learner
Hits Api endpoint with GET request
/userlearnergroupshandler/
Redirected to the handler
LearnerGroupsDashboardHandler
group_ids
learner_group_services.get_learner_groups_multi()
group_ids
Interacts with LearnerGroupDataModel and fetches learner groups maching the group_ids

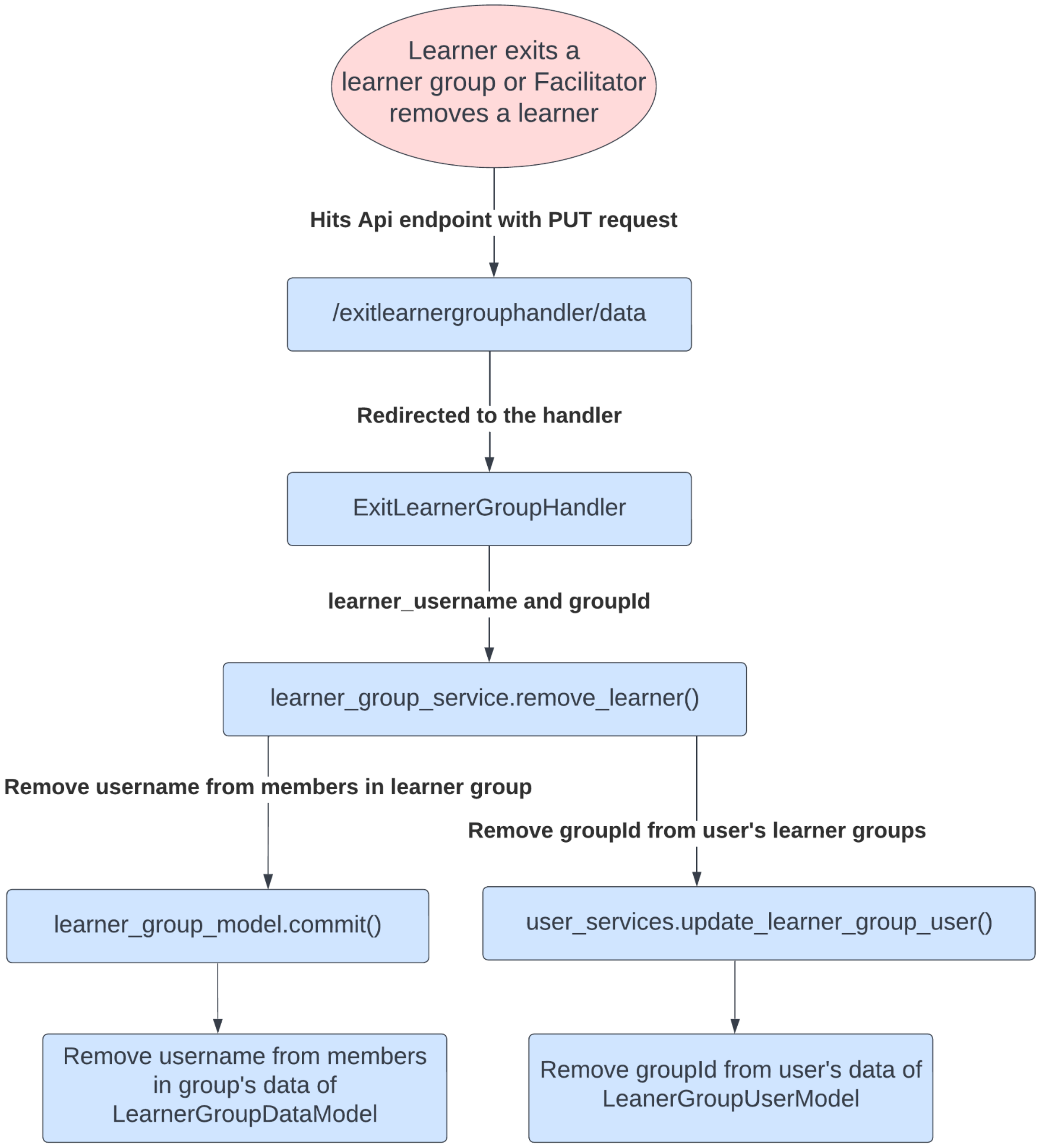
Accepting or Rejecting a learner group invitation:

- POST Request workflow when the learner accepts or rejects an invitation to join a learner group:


Learner accepts/declines a group invitation
Learner's action
Accept
Decline
/joinlearnergrouphandler/data
/declinelearnergroupinvitationhandler/data
Redirected to the handler
Redirected to the handler
LearnerGroupInvitationHandler
username and group Id
learner_group_service.update_invitations()
Check if invitation was accepted
True
False
Add username to learner group members, remove it from it's invitations, add groupId to user's learner groups and remove it from user's invitations
Remove username from learner group's invitations and remove groupId from user's invitations
learner_group_model.commit()
user_services.update_learner_group_user()
learner_group_model.commit()
Remove username from invitations and add username to members in group's data of LearnerGroupDataModel
Remove groupId from learner_group_invitaions in user's data of LeanerGroupUserModel
Remove username from invitations in group's data of LearnerGroupDataModel

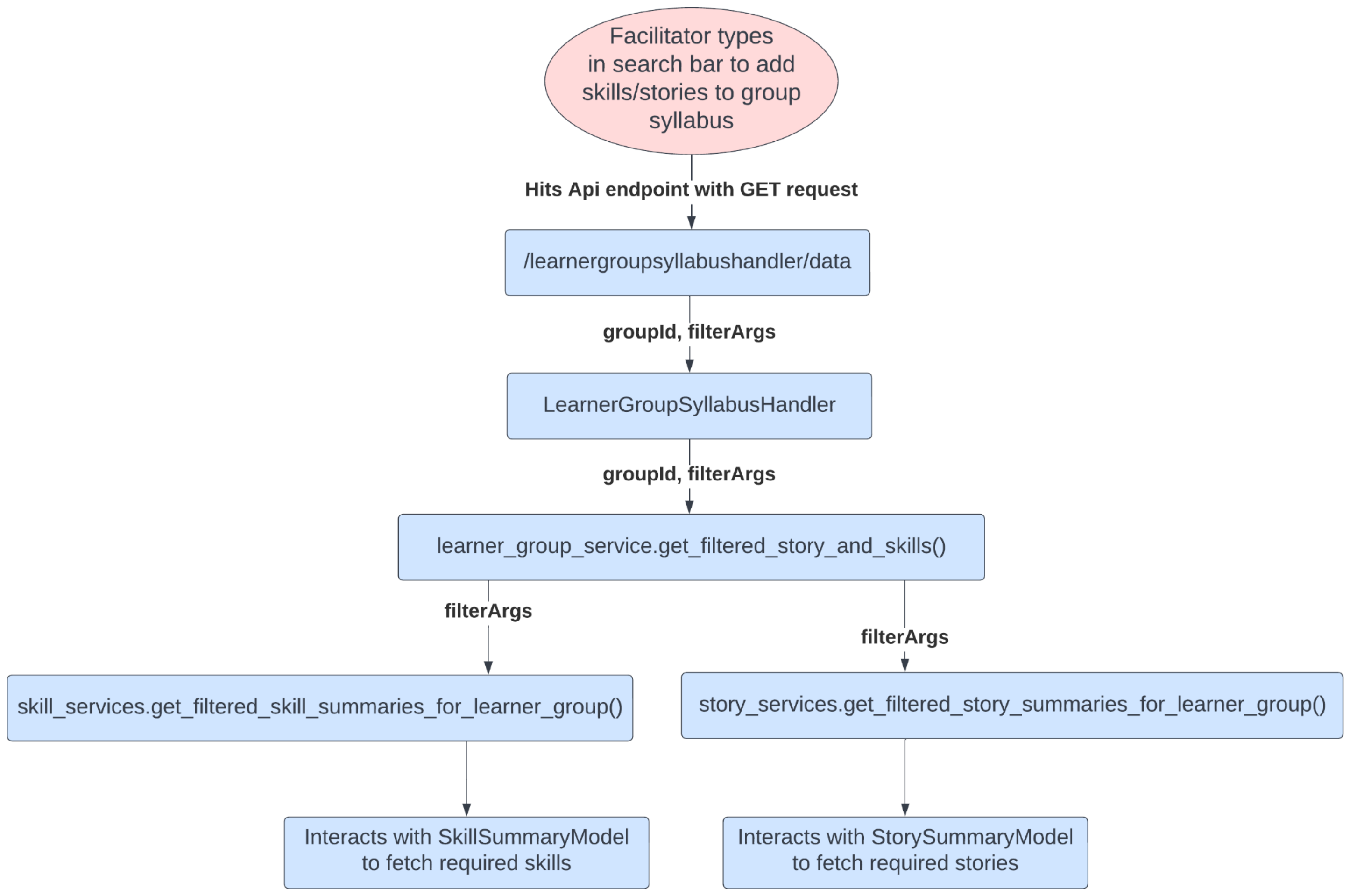
Leaving a learner group:

- POST Request workflow when a learner leaves a learner group:


Learner exits a learner group or Facilitator removes a learner
Hits Api endpoint with PUT request
/exitlearnergrouphandler/data
Redirected to the handler
ExitLearnerGroupHandler
learner_username and groupId
learner_group_service.remove_learner()
Remove username from members in learner group
Remove groupId from user's learner groups
learner_group_model.commit()
user_services.update_learner_group_user()
Remove username from members in group's data of LearnerGroupDataModel
Remove groupId from user's data of LeanerGroupUserModel

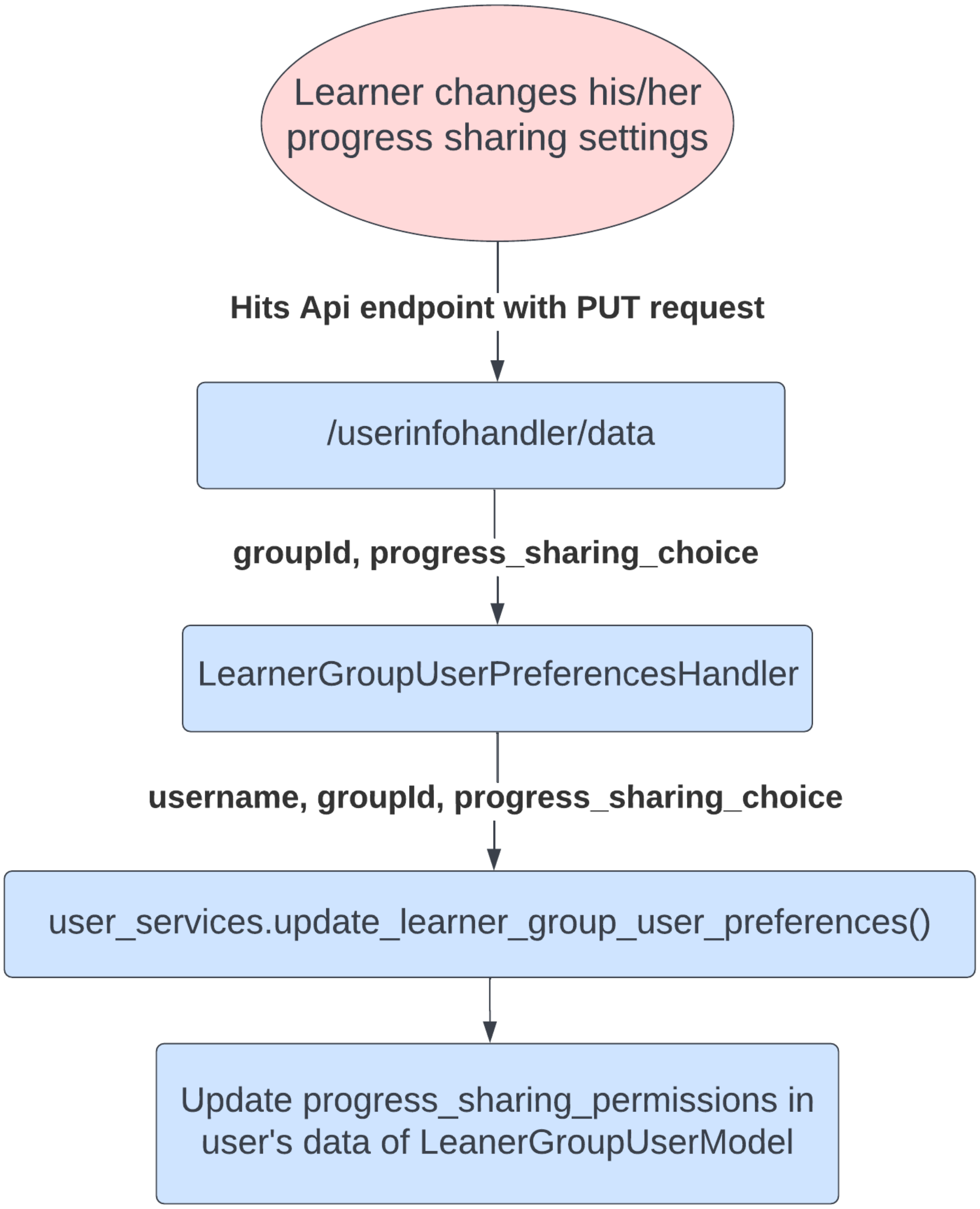
Selecting syllabus:

- GET Request workflow when the facilitator selects stories/skills to add to the learner group's syllabus:


Facilitator types in search bar to add skills/stories to group syllabus
Hits Api endpoint with GET request
/learnergroupsyllabushandler/data
groupId, filterArgs
LearnerGroupSyllabusHandler
groupId, filterArgs
learner_group_service.get_filtered_story_and_skills()
filterArgs
filterArgs
skill_services.get_filtered_skill_summaries_for_learner_group()
story_services.get_filtered_story_summaries_for_learner_group()
Interacts with SkillSummaryModel to fetch required skills
Interacts with StorySummaryModel to fetch required stories

Learner updating their group preferences:

- POST Request workflow when the learners change their progress sharing permissions :


Learner changes his/her progress sharing settings
Hits Api endpoint with PUT request
/userinfohandler/data
groupId, progress_sharing_choice
LearnerGroupUserPreferencesHandler
username, groupId, progress_sharing_choice
user_services.update_learner_group_user_preferences()
Update progress_sharing_permissions in user's data of LeanerGroupUserModel

Note: Workflow for updating group preferences by facilitator is already covered in Updating of learner group

Third-Party Libraries

No third party libraries are required for this project

“Service” Dependencies

No service dependencies.

Impact on Other Oppia Teams

It’s a new feature altogether. We will have to market it well to our learners and bring in teachers who are willing to guide our learners.

Key High-Level and Architectural Decisions

Decision 1: Fetching learner groups on teacher dashboard page

We have considered the following alternatives:

1. Get all learner groups from **LearnerGroupDataModel** and check if the facilitator list of the learner group matches has user_id of the user viewing the teacher dashboard page.
2. Store owner_of_learner_groups in **LearnerGroupUserModel** separately corresponding to the current user’s user_id and directly fetch all learner groups to be displayed on the teacher dashboard page.

Among these, we believe that 1st is the best approach, because:

- In the first approach we can fetch all the required learner groups using the query function in gae models which will take a single db call whereas in the 2nd scenario first we would have to get the LearnerGroupUserData model and then the required learner groups by getting the LearnerGroupData models, which will make it 2 db calls.

The above approaches are contrasted in detail in the following table:

	Alternative 1	**Alternative 2**
Speed of fetching required learner groups	O(k) assuming k time for a db call	2*O(k) assuming k time for a db call
Space requirements	No extra space	Requires extra space for storing owner_of_learner_groups field

Risks and mitigations

No potential risks.

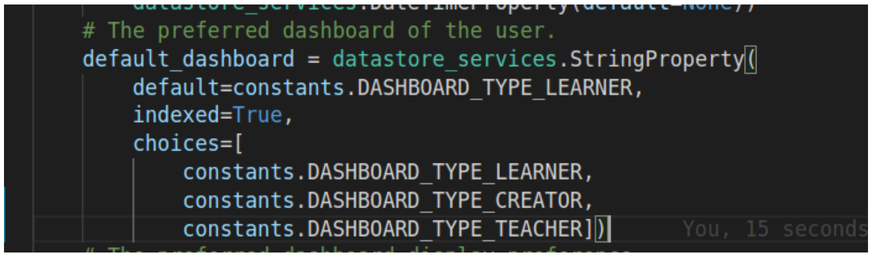
Implementation Approach

[Web only] Storage Model Layer Changes

Following are the new changes or modifications required at storage level:

- Inside /core/storage/learner_group
  - gae_models: Create a new storage model called **LearnerGroupDataModel**. It will store the following details
    - id: Id of the model indicating the id of the learner group. It's a 12 digit code made of ascii letters([a-z, A-Z]).
    - title: Name of the learner group
    - description: Description of the learner group
    - facilitators: List user ids of the facilitator of the learner group
    - members: List of user ids of all the members of the learner group
    - invitations: List of user ids of all the invitations sent by the facilitator to learners to join the group.
    - subtopics_ids: List of subtopic ids that are part of the group syllabus. Each subtopic id is stored as topicid**:**subtopicid string.
    - story_ids: List of story ids that are part of the group syllabus
  - Deletion policy: Model contains data to be deleted corresponding to a user: members, invitations. If the user being deleted is a learner then we just remove the user_id from above mentioned fields but if it is the facilitator we delete the whole learner group data model, hence it will be base_models.DELETION_POLICY.DELETE
  - Takeout Policy:
    - get_model_association_to_user(): Model contains user data for more than one user, hence it will be base_models.MODEL_ASSOCIATION_TO_USER.ONE_INSTANCE_SHARED_ACROSS_USERS.
    - get_export_policy(): Model contains data directly corresponding to a user hence the whole model data will be exported.
- Inside /core/storage/user
  - gae_models: Create a new storage model called **LearnerGroupUserModel**. It will store the following details
    - learner_group_invitations: List of group ids the learner is invited to join.
    - learner_groups: List of group ids the learner is part of.

        - progress_sharing_permissions: A map of groupId and true/false indicating the progress sharing permissions are true/false.
    - Deletion policy: Model contains data to delete corresponding to a user so the whole model is to be deleted for deletion , hence it will be base_models.DELETION_POLICY.DELETE
    - Takeout Policy:
        - get_model_association_to_user(): Model contains user data and it has one instance per user hence it will be base_models.MODEL_ASSOCIATION_TO_USER.ONE_INSTANCE_PER_USER.
        - get_export_policy(): Model contains data directly corresponding to a user hence the whole model data will be exported.
    - gae_models: Modify default_dashboard parameter of **UserSettingsModel** to accommodate option to set Teacher Dashboard Page as default dashboard page.



```
    # The preferred dashboard of the user.
    default_dashboard = datastore_services.StringProperty(
        default=constants.DASHBOARD_TYPE_LEARNER,
        indexed=True,
        choices=[
            constants.DASHBOARD_TYPE_LEARNER,
            constants.DASHBOARD_TYPE_CREATOR,
            constants.DASHBOARD_TYPE_TEACHER])      You, 15 seconds
```

Domain Objects

- learner_group_domain.py: Defines domain objects for handling learner group related data
    - LearnerGroup: It's the domain object for handling learner group data. It will have the following methods:
        - __init__(): Initializes a Learner Group domain object having the following arguments :
            - group_id: str. The id of the learner group.
            - title: str. The title of the learner group.
            - description: str. The description of the learner group.
            - facilitators: List user ids of the facilitator of the learner group
            - members: List of user ids of all the members of the learner group
            - invitations: List of user ids of all the invitations sent by the facilitator to learners to join the group.
            - subtopics_ids: List of subtopic ids that are part of the group syllabus. Each subtopic id is stored as topicid:subtopicid string.

            - story_ids: List of story ids that are part of the group syllabus

        - validate(): To validate domain object

    - LearnerGroupUser: It's the domain object for handling learner group user's details. It will have the following methods:
        - __init__(): Initializes a Learner Group User domain object having the following arguments :
            - user_id: str. The unique ID of the user.
            - invited_to_learner_groups: list(str). List of learner group ids that the user has been invited to.
            - member_of_learner_groups: list(str). List of learner group ids that the user is a member of.
            - progress_sharing_permissions: list(ProgressSharingPermissionsDict). Progress sharing permissions of all learner groups the user is member of.

        - validate(): To validate domain object

    - ProgressSharingPermissionsDict: Dictionary representation of user's progress sharing permissions for groupIds. It will have the following arguments :
        - subtopic_id: str. It is depicted as topicid:subtopicid string
        - progress_in_skills: List[skill_domain.UserSkillMastery]

User Flows (Controllers and Services)

- learner_group.py(controller): Create a new learner group controller which will have the following handlers:
    - LearnerGroupHandler:
        - Called on POST request from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/create/data** from createNewLearnerGroupAsync() it calls create_group(title, description, invitations, syllabus) from learner_group_services which internally creates a new learner group by interacting with **LearnerGroupDataModel** and **LearnerGroupUserModel**.
          It extracts learner group's title, description, invitations list and syllabus dict from the request body and creates a learner group model from it, and returns a LearnerGroup domain object as dictionary.

        - Called on PUT request from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/update/data** from updateLearnerGroupAsync() it calls update_group(group_id, title,

description, invitations, syllabus) from learner_group_services which internally updates the learner group by interacting with **LearnerGroupDataModel** and **LearnerGroupUserModel**.
It extracts learner group's id(which is used to get the learner group to update), title, description, invitations list and syllabus dict from the request body and updates the learner group model with new values, and returns a LearnerGroup domain object as dictionary.

    - Called on DELETE request from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/delete/<groupId>** from deleteLearnerGroupAsync(), it calls delete_group(group_id) from learner_group_services which internally deletes the learner group by interacting with **LearnerGroupDataModel** and **LearnerGroupUserModel**. It takes the learner group's id as delete request parameter, deletes the learner group and returns a LearnerGroup domain object of the deleted group as a dictionary.

    - Called on GET request from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/<groupId>** from getLearnerGroupInfoAsync(), it gets group_id as request parameter which is used to fetch the associated learner group data. It then checks if the current user's user_id is the same as the facilitator's user_id.
      If it matches, it calls get_facilitator_view_of_learner_group(group_id) from learner_group_services and if it does not match, it checks if the current user's user_id is present in members list of the learner group, it it is present it calls get_learner_view_of_learner_group(group_id) else it exits the handler returning an error message.
      For valid user_id after checking the handler returns a list of objects as dictionaries which have learner's info and learner's progress through group syllabus. It will be a single element list for the learner's view and multiple elements list for the facilitator's view.

- TeacherDashboardHandler:
    - Called on GET request from **teacher-dashboard-backend-api-service** at endpoint **/teacherdashboardhandler/** from getTeacherDashboardLearnerGroupsAsync(), the get request does not accept any parameters, it fetches and stores all the learner groups of the facilitator as owner_learner_groups by calling get_learner_groups_with_role_as_facilitator() which gets the corresponding LearnerUserDataModel of the user and returns owner_of_learner_groups field from it which returns a list of group_ids, these group ids are passed to get_learner_groups_multi() from

learner_group_services which returns a list of LearnerGroup domain objects which is then returned by the handler.

- LearnerGroupInvitationHandler:
    - If the learner accepts the learner group’s invitation, a PUT request is made from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/join/data** from joinLearnerGroupAsync(). It extracts group_id, has_accepted_invitation from request body and then calls update_invitation(group_id, username, has_accepted_invitation=True) from learner_group_services which internally removes invitation and adds the user_id of the current user as members by interacting with **LearnerGroupDataModel** and **LearnerGroupUserModel** and returns a success or failure message based on if the update was successful or not. This message is then also returned by the handler.

    - If the learner rejects the learner group’s invitation, a PUT request is made from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/decline/data** from rejectLearnerGroupInvitationAsync(), It extracts group_id, has_accepted_invitation from request body and then calls update_invitation(group_id, username, has_accepted_invitation=False) from learner_group_services which internally removes current user’s user_id from invitations of learner group and the group_id from learner_group_invitations field of the associated learner group user data model. The function returns a success or failure message based on if the update was successful or not. This message is then also returned by the handler.

- ExitLearnerGroupHandler:
    - Called on PUT request from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/exit/data/** from exitLearnerGroupAsync(). It extracts group_id, username from the request body and then calls remove_learner(group_id, username) from learner_group_services which internally removes the learner from learner group by interacting with **LearnerGroupDataModel** and **LearnerGroupUserModel**. The function returns a success or failure message based on if the removal was successful or not. This message is then also returned by the handler.

- LearnerGroupSyllabusHandler:
    - Called on GET request from **learner-group-backend-api-service** at endpoint **/learnergrouphandler/syllabus/data/** from getLearnerGroupSyllabusItemsAsync(). It extracts group_id and filter_args

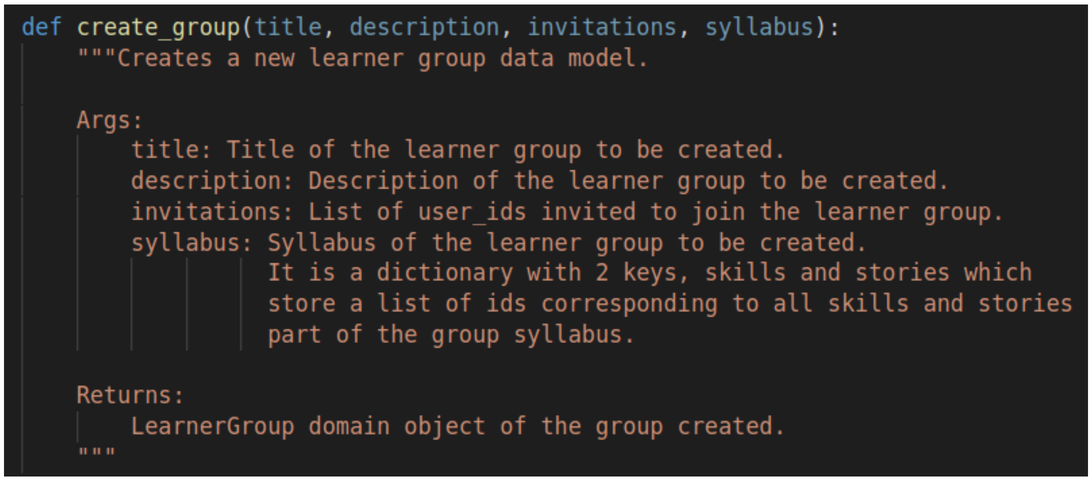
    from the request body and calls get_filtered_story_and_skills(group_id, filter_args) which filters the stories and skills based on filter arguments and returns a LearnerGroupSyllabus domain object, this domain object is then returned by the handler.

- LearnerGroupUserPreferencesHandler:
    - Called on PUT request from **learner-group-backend-api-service** at endpoint **/userinfohandler/data/** from updateLearnersGroupPreferencesAsync(). It extracts group_id and progress_sharing_choice from the request body and calls update_learner_group_user_preferences(group_id, username, progress_sharing_choice) from user_services which internally updates learner's progress_sharing_permissions of the learner group corresponding to the provided group_id in the LearnerGroupUserModel. The handler returns a success or failure message based on success or failure of updating learner's progress sharing permissions.

- LearnerGroupsDashboardHandler:
    - On GET request from **learner-group-backend-api-service** at endpoint **/learnergroupsdashboardhandler/** from getAllLearnerGroupsOfUserAsync(), it gets the username of the current user and fetches the **LearnerGroupUserModel** for that user and gets info of all invited learner groups and learner groups that the learner is part of by calling get_learner_groups_multi(user_id) from learner_group_services which creates and returns a list of LearnerGroup domain objects.
      The handler returns a dictionary with 'invited_groups' and 'learner_groups' as keys and list of LearnerGroup domain objects as values representing the info of learner groups the learner is invited in and the ones the learner has joined.

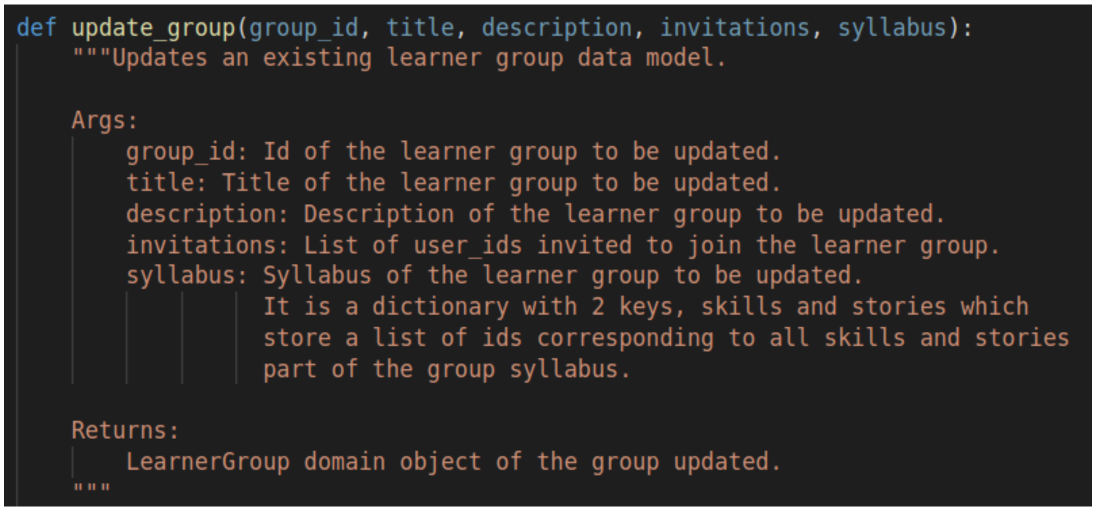
- learner_group_services.py(service): Create a new learner group service which will have the following functions:

- create_group():



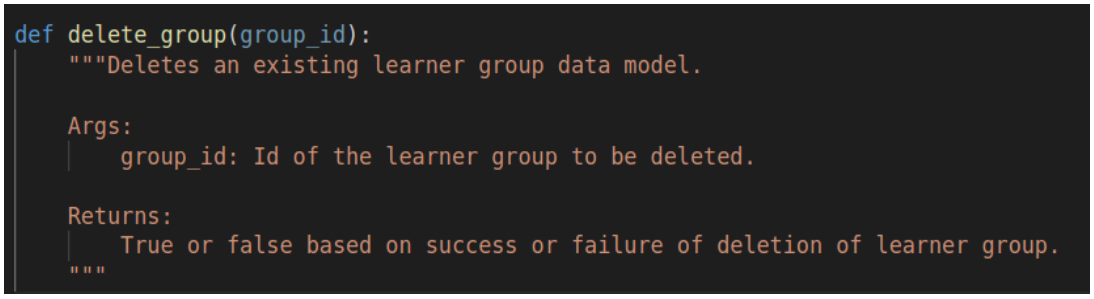
```python
def create_group(title, description, invitations, syllabus):
    """Creates a new learner group data model.

    Args:
        title: Title of the learner group to be created.
        description: Description of the learner group to be created.
        invitations: List of user_ids invited to join the learner group.
        syllabus: Syllabus of the learner group to be created.
                    It is a dictionary with 2 keys, skills and stories which
                    store a list of ids corresponding to all skills and stories
                    part of the group syllabus.

    Returns:
        LearnerGroup domain object of the group created.
    """
```

- update_group():



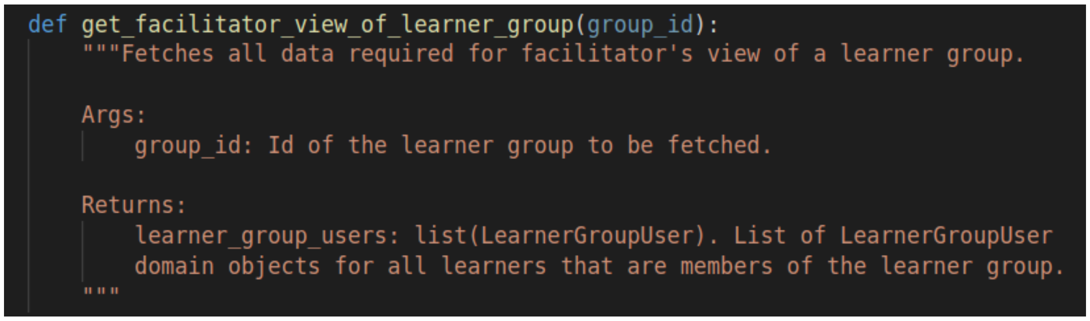
```python
def update_group(group_id, title, description, invitations, syllabus):
    """Updates an existing learner group data model.

    Args:
        group_id: Id of the learner group to be updated.
        title: Title of the learner group to be updated.
        description: Description of the learner group to be updated.
        invitations: List of user_ids invited to join the learner group.
        syllabus: Syllabus of the learner group to be updated.
                    It is a dictionary with 2 keys, skills and stories which
                    store a list of ids corresponding to all skills and stories
                    part of the group syllabus.

    Returns:
        LearnerGroup domain object of the group updated.
    """
```

- delete_group():



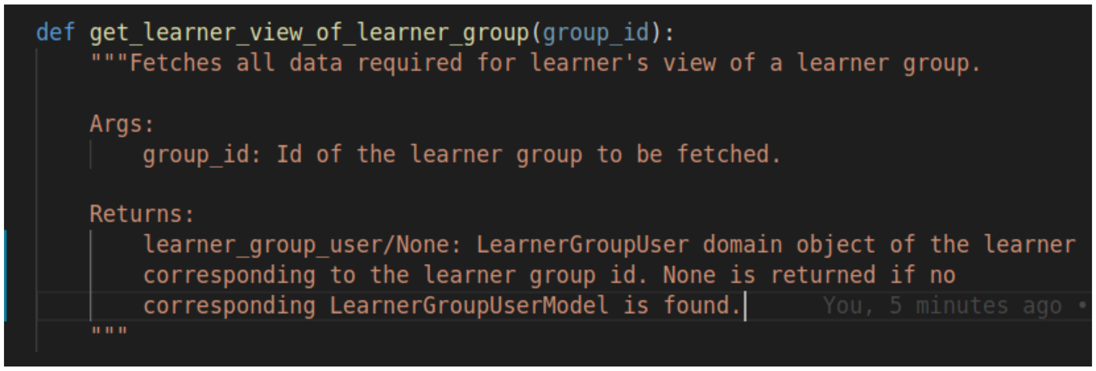
```python
def delete_group(group_id):
    """Deletes an existing learner group data model.

    Args:
        group_id: Id of the learner group to be deleted.

    Returns:
        True or false based on success or failure of deletion of learner group.
    """
```

- get_facilitator_view_of_learner_group():



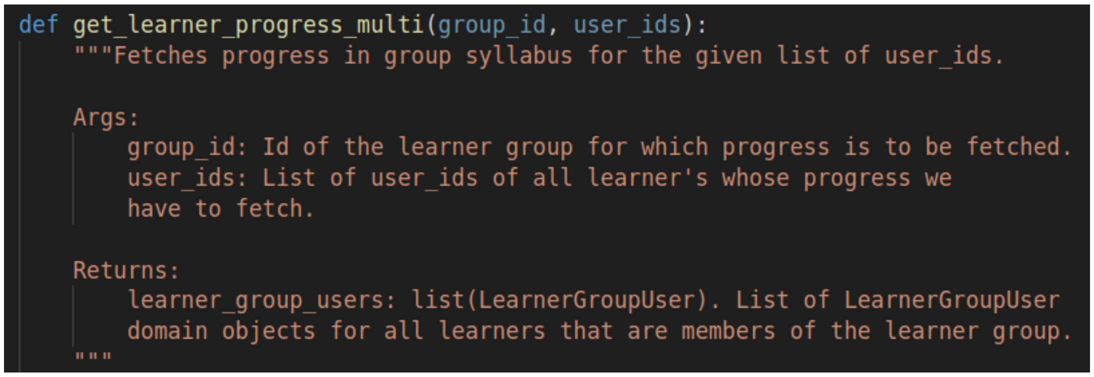
```
def get_facilitator_view_of_learner_group(group_id):
    """Fetches all data required for facilitator's view of a learner group.

    Args:
        group_id: Id of the learner group to be fetched.

    Returns:
        learner_group_users: list(LearnerGroupUser). List of LearnerGroupUser
        domain objects for all learners that are members of the learner group.
    """
```

- get_learner_view_of_learner_group():



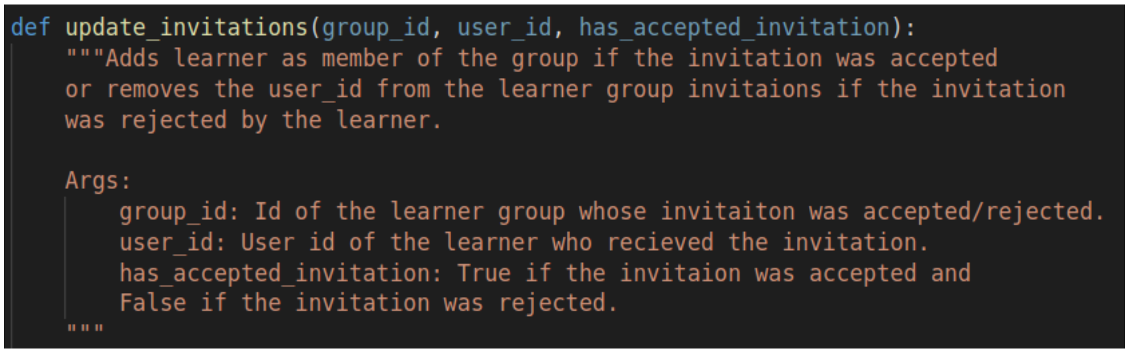
```
def get_learner_view_of_learner_group(group_id):
    """Fetches all data required for learner's view of a learner group.

    Args:
        group_id: Id of the learner group to be fetched.

    Returns:
        learner_group_user/None: LearnerGroupUser domain object of the learner
        corresponding to the learner group id. None is returned if no
        corresponding LearnerGroupUserModel is found.      You, 5 minutes ago •
    """
```

- get_learner_progress_multi():



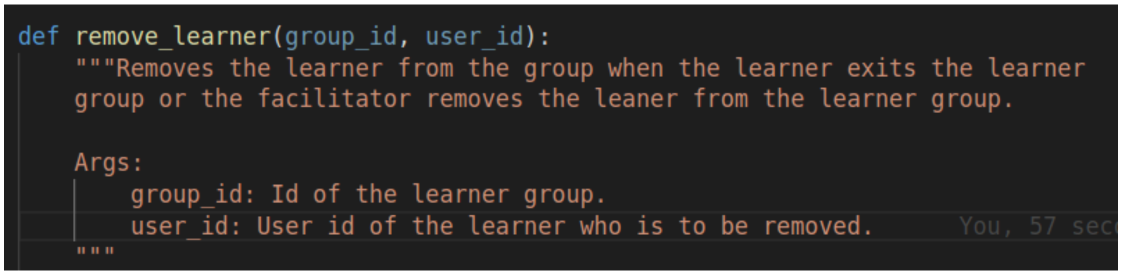
```
def get_learner_progress_multi(group_id, user_ids):
    """Fetches progress in group syllabus for the given list of user_ids.

    Args:
        group_id: Id of the learner group for which progress is to be fetched.
        user_ids: List of user_ids of all learner's whose progress we
        have to fetch.

    Returns:
        learner_group_users: list(LearnerGroupUser). List of LearnerGroupUser
        domain objects for all learners that are members of the learner group.
    """
```

- update_invitation():



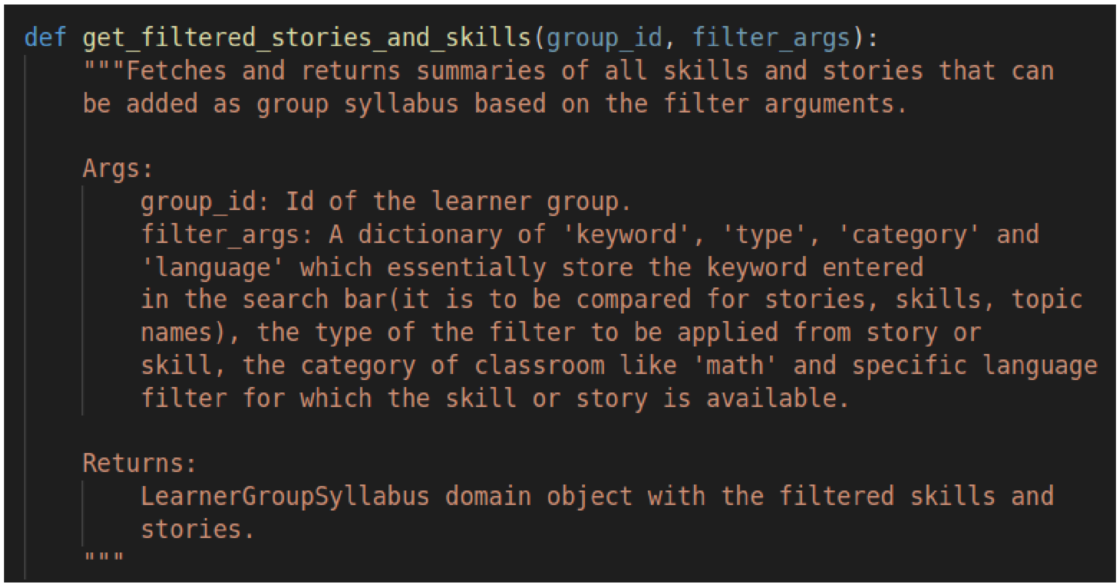
```python
def update_invitations(group_id, user_id, has_accepted_invitation):
    """Adds learner as member of the group if the invitation was accepted
    or removes the user_id from the learner group invitaions if the invitation
    was rejected by the learner.

    Args:
        group_id: Id of the learner group whose invitaiton was accepted/rejected.
        user_id: User id of the learner who recieved the invitation.
        has_accepted_invitation: True if the invitaion was accepted and
        False if the invitation was rejected.
    """
```

- remove_learner():



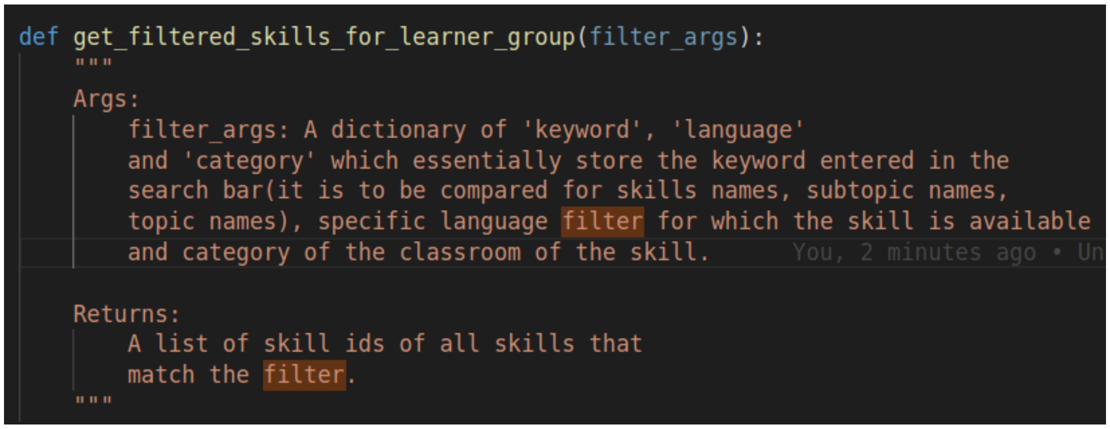
```python
def remove_learner(group_id, user_id):
    """Removes the learner from the group when the learner exits the learner
    group or the facilitator removes the leaner from the learner group.

    Args:
        group_id: Id of the learner group.
        user_id: User id of the learner who is to be removed.       You, 57 sec
    """
```

- get_filtered_story_and_skills():



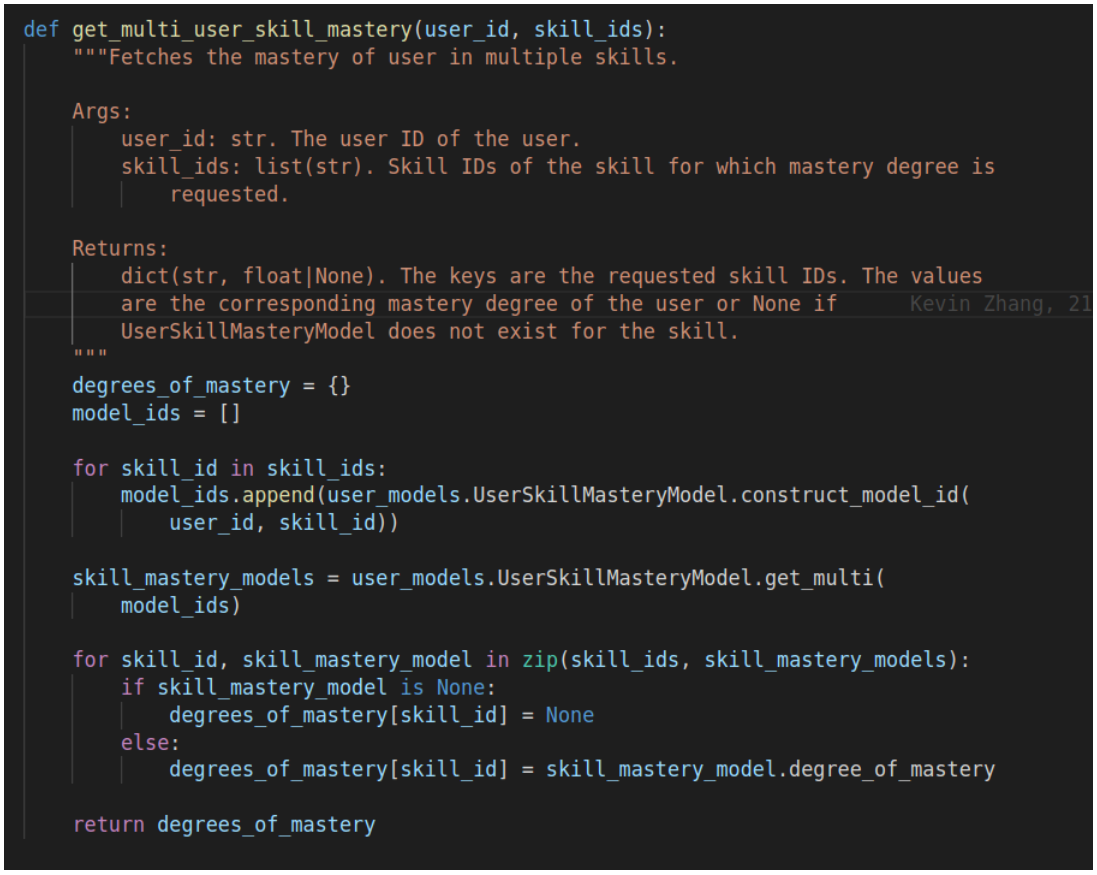
```python
def get_filtered_stories_and_skills(group_id, filter_args):
    """Fetches and returns summaries of all skills and stories that can
    be added as group syllabus based on the filter arguments.

    Args:
        group_id: Id of the learner group.
        filter_args: A dictionary of 'keyword', 'type', 'category' and
        'language' which essentially store the keyword entered
        in the search bar(it is to be compared for stories, skills, topic
        names), the type of the filter to be applied from story or
        skill, the category of classroom like 'math' and specific language
        filter for which the skill or story is available.

    Returns:
        LearnerGroupSyllabus domain object with the filtered skills and
        stories.
    """
```

- skill_services.py(service): Update skill service to have the following functions:
  - get_filtered_skill_summaries_for_learner_group():



```
def get_filtered_skills_for_learner_group(filter_args):
    """
    Args:
        filter_args: A dictionary of 'keyword', 'language'
        and 'category' which essentially store the keyword entered in the
        search bar(it is to be compared for skills names, subtopic names,
        topic names), specific language filter for which the skill is available
        and category of the classroom of the skill.      You, 2 minutes ago • Un

    Returns:
        A list of skill ids of all skills that
        match the filter.
    """
```

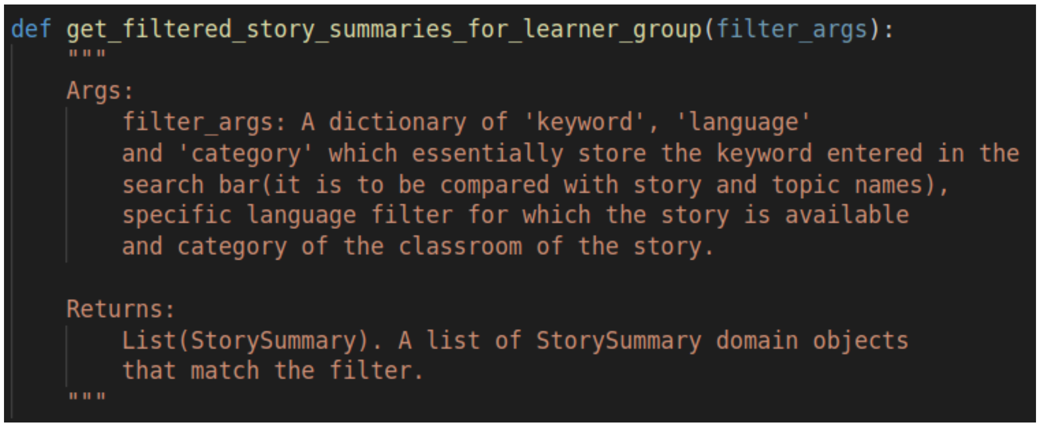
  - get_multi_user_skill_mastery(): Already exists, no changes are required here.



```
def get_multi_user_skill_mastery(user_id, skill_ids):
    """Fetches the mastery of user in multiple skills.

    Args:
        user_id: str. The user ID of the user.
        skill_ids: list(str). Skill IDs of the skill for which mastery degree is
            requested.

    Returns:
        dict(str, float|None). The keys are the requested skill IDs. The values
        are the corresponding mastery degree of the user or None if      Kevin Zhang, 21
        UserSkillMasteryModel does not exist for the skill.
    """
    degrees_of_mastery = {}
    model_ids = []

    for skill_id in skill_ids:
        model_ids.append(user_models.UserSkillMasteryModel.construct_model_id(
            user_id, skill_id))

    skill_mastery_models = user_models.UserSkillMasteryModel.get_multi(
        model_ids)

    for skill_id, skill_mastery_model in zip(skill_ids, skill_mastery_models):
        if skill_mastery_model is None:
            degrees_of_mastery[skill_id] = None
        else:
            degrees_of_mastery[skill_id] = skill_mastery_model.degree_of_mastery

    return degrees_of_mastery
```

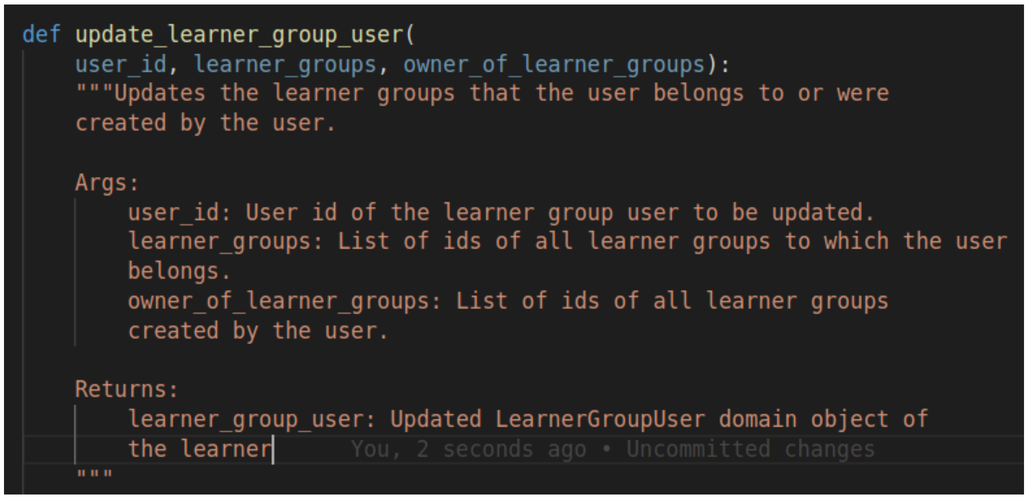
- story_services.py(service): Update story service to have the following functions:

- get_filtered_story_summaries_for_learner_group():



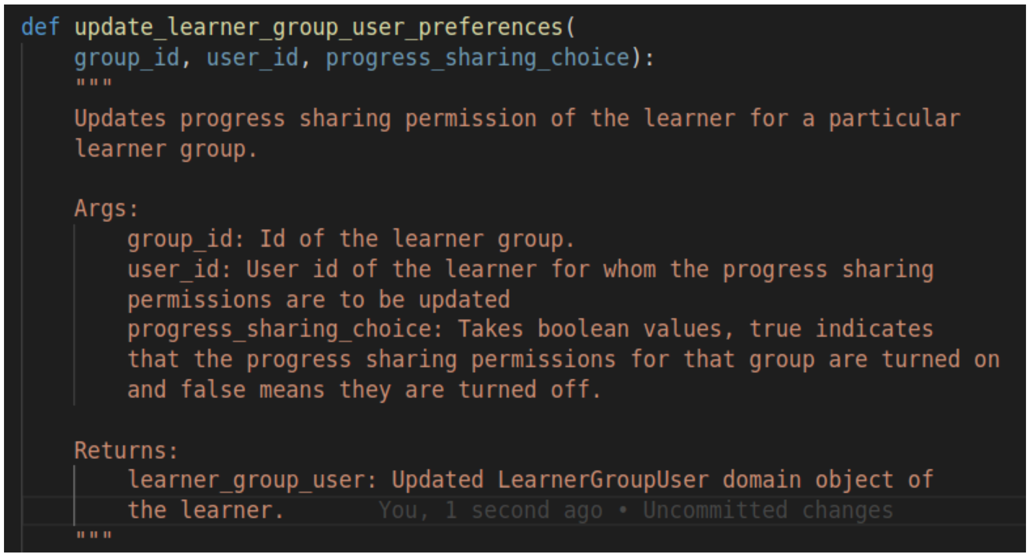
```
def get_filtered_story_summaries_for_learner_group(filter_args):
    """
    Args:
        filter_args: A dictionary of 'keyword', 'language'
        and 'category' which essentially store the keyword entered in the
        search bar(it is to be compared with story and topic names),
        specific language filter for which the story is available
        and category of the classroom of the story.

    Returns:
        List(StorySummary). A list of StorySummary domain objects
        that match the filter.
    """
```

- user_services.py(service): Update user service to have the following functions:
  - update_learner_group_user():



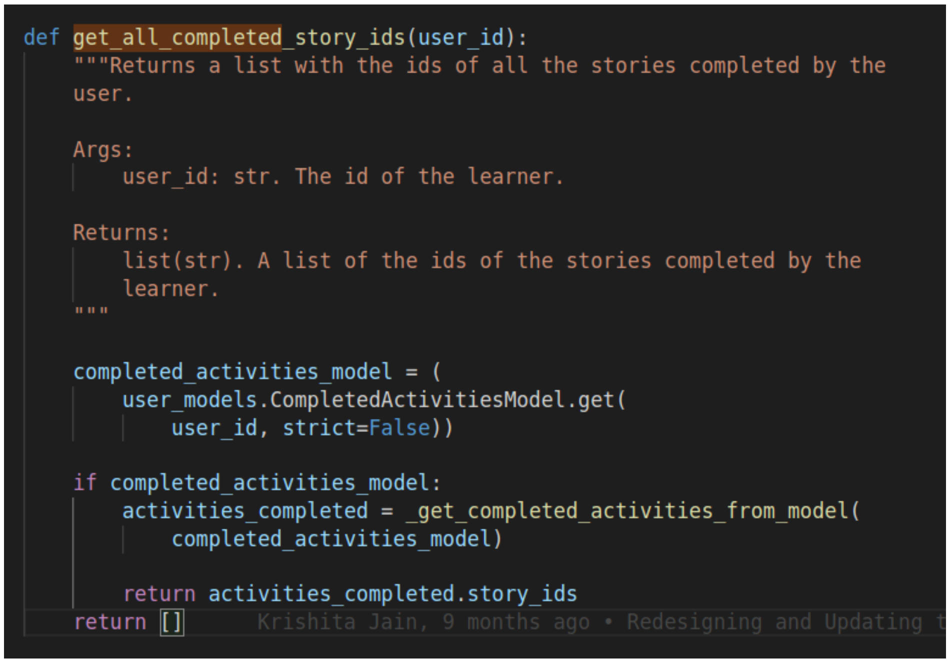
```
def update_learner_group_user(
    user_id, learner_groups, owner_of_learner_groups):
    """Updates the learner groups that the user belongs to or were
    created by the user.

    Args:
        user_id: User id of the learner group user to be updated.
        learner_groups: List of ids of all learner groups to which the user
        belongs.
        owner_of_learner_groups: List of ids of all learner groups
        created by the user.

    Returns:
        learner_group_user: Updated LearnerGroupUser domain object of
        the learner      You, 2 seconds ago • Uncommitted changes
    """
```

  - update_learner_group_user_preferences():



```
def update_learner_group_user_preferences(
    group_id, user_id, progress_sharing_choice):
    """
    Updates progress sharing permission of the learner for a particular
    learner group.

    Args:
        group_id: Id of the learner group.
        user_id: User id of the learner for whom the progress sharing
        permissions are to be updated
        progress_sharing_choice: Takes boolean values, true indicates
        that the progress sharing permissions for that group are turned on
        and false means they are turned off.

    Returns:
        learner_group_user: Updated LearnerGroupUser domain object of
        the learner.      You, 1 second ago • Uncommitted changes
    """
```

- learner_progress_services.py(service): No changes required, we are only using this service.
    - get _all_completed_story_ids(): This function already exists.



```
def get_all_completed_story_ids(user_id):
    """Returns a list with the ids of all the stories completed by the
    user.

    Args:
        user_id: str. The id of the learner.

    Returns:
        list(str). A list of the ids of the stories completed by the
        learner.
    """

    completed_activities_model = (
        user_models.CompletedActivitiesModel.get(
            user_id, strict=False))

    if completed_activities_model:
        activities_completed = _get_completed_activities_from_model(
            completed_activities_model)

        return activities_completed.story_ids
    return []       Krishita Jain, 9 months ago • Redesigning and Updating t
```

[Web only] Web frontend changes

Frontend Models:

- LearnerGroup Model: Model for storing learner group objects in the frontend.
    - It will have the following variables:
        - groupId: string: Id of the learner group
        - title: string: Title/Name of the learner group
        - description: string: Description of the learner group
        - invitations: UserInfo[ ]: Info about the users to whom the invitations have been sent to
        - members: LearnerGroupUser[ ]: Info about the users along with their progress through the group syllabus
        - syllabus: LearnerGroupSyllabus[ ]: Info about all the syllabus items currently present in the group.

    - It will further have getter methods and a createFromBackendDict(learnerGroupBackendDict) which takes in argument learnerGroupBackendDict fetched by the get api calls and converts it to the LearnerGroup model object which can be used in the frontend.

- LearnerGroupUser Model: Model for storing learner group user objects in the frontend. Stores info about the user and their progress through the group syllabus
    - It will have the following variables:
        - username: string: Username to be displayed in the frontend of the user
        - progressInSubtopics: UserProgressInSubtopic[ ]: List of progress the user has made in the subtopics. It further consists of subtopic name and skill masteries of the user
        - progressInStories: StorySummary[ ]: List of progress of the user made in the stories.

    - It will further have getter methods and a createFromBackendDict(learnerGroupUserBackendDict) which takes in argument learnerGroupUserBackendDict fetched by the get api calls and converts it to the LearnerGroupUser model object which can be used in the frontend.

- LearnerGroupSyllabus Model: Model for storing learner group syllabus objects in the frontend. Stores info about stories and the subtopic(combination of skills) that are part of the group syllabus.
    - It will have the following variables:
        - subtopicSummaries: LearnerGroupSubtopicSummary[ ]: Subtopic summaries of all subtopics part of the syllabus. LearnerGroupSubtopicSummary object further includes the following variables:
            - title
            - skillIds
            - thumbnailFilename
            - thumbnailBgColor
            - urlFragment
        - storySummaries: LearnerGroupStorySummary[ ]: Summaries of the stories that are part of the learner group. LearnerGroupStorySummary object further includes the following variables:
            - storyId
            - title
            - description
            - languageCode
            - thumbnailFilename
            - thumbnailBgColor
            - urlFragment
            - storyNodes
            - topicId
            - topicName
            - topicUrlFragment

            - classroomUrlFragment
    - It will further have getter methods and a createFromBackendDict(learnerGroupSyllabusBackendDict) which takes in argument learnerGroupSyllabusBackendDict fetched by the get api calls and converts it to the LearnerGroupSyllabus model object which can be used in the frontend.

Frontend Services:

- teacher-dashboard-backend-api.service.ts: New service that makes api requests to fetch learner groups from backend for teacher dashboard page.
    - This service will have a function called getTeacherDashboardLearnerGroupsAsync() to fetch all learner groups to be shown on the teacher dashboard page. The function does not accept any argument but returns a promise resolving to a list of LearnerGroup model objects.
- learner-group-backend-api.service.ts: The service will have functions for fetching and storing learner group related data from and to the backend.
    - It will have all functions specified earlier connecting with handlers though API endpoints.
    - It will have the following functions:
        - createNewLearnerGroupAsync():
            - Arguments: learnerGroupInfo: LearnerGroup model object
            - Returns: Promise resolving to LearnerGroup model object of the newly created learner group.
        - updateLearnerGroupAsync():
            - Arguments: learnerGroupInfo: LearnerGroup model object with updated values
            - Returns: Promise resolving to LearnerGroup model object of the updated learner group.
        - deleteLearnerGroupAsync():
            - Arguments: groupId: Id of the learner group to delete
            - Returns: Promise resolving to LearnerGroup model object of the deleted learner group.
        - getLearnerGroupInfoAsync():
            - Arguments: groupId: Id of the learner group to get info of
            - Returns: Promise resolving to LearnerGroup model object of the requested learner group.
        - getTeacherDashboardLearnerGroupsAsync():
            - Arguments: No arguments
            - Returns: Promise resolving to list of LearnerGroup model objects that facilitator is owner of.
        - joinLearnerGroupAsync():

            - Arguments: groupId: Id of the learner group to join
            - Returns: Promise resolving to a boolean value signifying if the request was successful or not.
        - rejectLearnerGroupInvitationAsync():
            - Arguments: groupId: Id of the learner group to reject invitation of.
            - Returns: Promise resolving to a boolean value signifying if the request was successful or not.
        - exitLearnerGroupAsync():
            - Arguments: groupId: Id of the learner group to exit, username: username of the user exiting the learner group.
            - Returns: Promise resolving to a boolean value signifying if the request was successful or not.
        - getLearnerGroupSyllabusItemsAsync():
            - Arguments: groupId: Id of the learner group for which the syllabus is to be fetched
            - Returns: Promise resolving to a list of LearnerGroupSyllabus model objects.
        - updateLearnersGroupPreferencesAsync():
            - Arguments: groupId: string, isProgressSharingTurnedOn: boolean
            - Returns: void type promise.
        - getAllLearnerGroupsOfUserAsync():
            - Arguments: No arguments
            - Returns: Promise resolving to invitatedLearnerGroups and learnerGroups containing list of LearnerGroup model objects.

Frontend Components:

- For Teacher Dashboard Page:
    - teacher-dashboard-page.component.ts: New component for displaying the teacher dashboard page.
        - This will show all learner groups that are created by the facilitator as cards.
        - To show all these learner groups, on initial load, it will call getTeacherDashboardLearnerGroupsAsync() to fetch required learner groups data from backend as described before.
        - All the learner groups fetched will be stored inside a list called allLearnerGroups which is a list of learnerGroup model. This will be later iterated over to create cards for each learner group.
        - It will also display a create learner group button which will open a modal for creating the learner group from learner-group-creation-modal component.

- For facilitator’s view:
    - learner-group-facilitator-view-page.component.ts: New component for displaying the facilitator's view of the learner group page.
        - On initial load when the url “/learner-groups/<groupId>” is hit, getGroupIdFromUrl() is called which gets the groupId from the url and stores it inside a groupId variable.
        - Once we have the groupId we fetch the learnerGroupInfo using the function getLearnerGroupInfoAsync(groupId) from learner-group-backend-api-service which returns all required info to display the learner group page.

    - learner-group-facilitator-preferences.component.ts
        - It will have an option to delete the learner group.
        - It will have 2 tabs, Group Details and Group Members
        - Group Details will have an option edit group title and group description
        - The Group Members tab will have options to add/remove members and revoke invitations.

    - learner-group-learner-specific-progress-tab.component.ts
        - Will show a brief info on progress of all members of the group as cards
        - For learners who have not turned on their progress sharing permissions, it will not permissions not given message inside the card
        - For learners who have turned on progress sharing permissions, a view details button will be shown which will redirect the learner onto the learner specific progress page.

    - learner-group-learner-specific-progress-page.component.ts
        - Will show progress of the selected learner
        - It has 2 tabs, Skill Analysis tab and Progress in stories tab which shows learner-group-skill-analysis-tab component and learner-group-progress-in-stories-tab component on click of the tab
        - By default Skill Analysis tab is opened

    - learner-group-skill-analysis-tab.component.ts
        - Shows skills progress of a specific learner in the syllabus skills
        - It has a variable called learnerSkillsMastery which stores a list of SkillMastery objects which consists of topic wise skill masteries from the skills present in the syllabus.
        - This learnerSkillsMastery list is iterated over to create the cards for all skills.

    - learner-group-progress-in-stories-tab.component.ts
        - Shows progress of a specific learner in the stories present in the syllabus items

    - learner-group-overview-tab.component.ts
        - Shows an overview of the learner group
        - It shows number of members in the group, number of stories and number of skills in the assigned syllabus of the group

    - learner-group-syllabus-tab.component.ts
        - It shows all syllabus items that are currently present in the learner group with an option to remove them.
        - It also shows a search bar with filters which can be used to add new items to the learner group syllabus.
    - learner-group-creation-modal.component.ts
        - It's the modal for creating a new group
        - It takes all inputs to create the learner group and calls createLearnerGroupAsync() with all info which creates a new learner group.

- For Learner's view:
    - learner-dashboard-learner-groups-tab.component.ts
        - It shows all the groups that the learner is part of and all the learner group invitations he/she has received.
        - On initial load of this page a GET request is sent using getAllLearnerGroupsOfUserAsync() from learner-group-backend-api-service which returns a dictionary of learnerGroups and learnerGroupInvitations.
        - The learnerGroups variable will store all the learner groups the learner is part of, learnerGroupInvitations will store all the learner groups for which the learner has received the invitations.
        - learnerGroupInvitations are used to create cards in the invitations section of this page.
        - learnerGroups are used to create cards in the learner groups section of this page
        - learnerGroups and learnerGroupInvitations are lists of LearnerGroup models which store all attributes of **LearnerGroupDataModel** in the frontend.

    - learner-group-learner-view-page.component.ts
        - It shows the learner's view of a specific learner group page.
        - On initial load we get the groupId by calling getGroupIdFromUrl() which splits and returns the groupId from the page url.

        - By default the home tab is selected on the page.
        - The page will have a group preferences button and a leave group button. On clicking the group preferences button the group preferences page is opened and on clicking the leave group button a confirmation box is shown asking learners if they want to leave the group.

    - learner-group-learner-preferences.component.ts
        - It will have 3 tabs: Group Details, Privacy Settings, Help and FAQ
        - Group Details tab will show all group members and the current user
        - Privacy Settings tab will have the option to switch progress sharing permissions.
        - Help and FAQ will show a fixed set of FAQs related to learner groups.

    - learner-group-learner-home-tab.component.ts
        - It shows the count of stories completed and skills mastered by the learner.

    - learner-group-learner-view-syllabus-tab.component.ts
        - Shows the progress of the learner in the assigned group syllabus.
        - It creates stories and skills cards by iterating over StoriesProgress and SkillsMastery lists and also provides a button to start practicing them.

    - learner-group-skill-proficiency-tab.component.ts
        - Shows skill mastery of the learner in the assigned group syllabus skills.
        - These skill masteries are shown per topic using the SkillsMastery list.

    - learner-group-invitation-modal.component.ts:
        - It will show a brief info about the learner group stored in groupTitle, groupDescription, membersCount. It will also have buttons to join the group and decline the invitation.
        - The modal component is called in the learner-dashboard-learner-groups-tab component when the view details button of a learner group is clicked.
        - The variables groupTitle, groupDescription, membersCount, groupId would be input type variables which will be passed from the learner-dashboard-learner-groups-tab component.

Documentation changes

The project does not have any development related changes, hence no documentation changes are required.

Testing Plan

E2e testing plan

#	Test name	Initial setup step	Step	Expectation
1.	Facilitator can create a learner group successfully	Login and open teacher dashboard page	On the teacher dashboard page click on create learner group button	A create learner group modal is opened
			In the opened leaner group modal add group name and description and click next button	Group syllabus tab is opened
			Select group syllabus and click next	Add Members tab is opened
			Add usernames that are to be invited to join the group and click create group button	The group is successfully created and the facilitator is redirected to the created learner group page.
2.	Facilitator can update learner group successfully	Login and open a previously created learner group	Go to the group preferences page of the learner group and update group details and group members and reload the page	Group details and group members show updated fields
			Go to the syllabus tab on the home learner group page and click add new syllabus item	Add new syllabus item page is opened
			Add a new syllabus item and click done.	Facilitator is redirected to the syllabus tab of the learner group page and the newly added syllabus items can be seen.
3.	Learner can be invited and can join/decline a learner group invitation	Login as facilitator and send invitation and logout, then login as learner.	As a facilitator of the learner group go to the group preferences page and add a new username to invite the group	An invitation is sent to the user and the invitations list shows the updated list of users
			Now login as the invited learner and from the invitations tab on the common learner groups page, accept the invitation	Learner successfully joins the group and is redirected to the learner group page of the group.
			Repeat the above steps again and decline the invitation	Invitation is rejected and removed from the invitations section of the common learner group page.

4.	Facilitator can only view the learner's progress if progress sharing permissions are turned on.	Create a learner group as facilitator and invite a learner to join the group	Login as the learner invited	Successfully logged in
			Go to the common learner groups page and accept the invitation.	
			Logout as the learner and login back as the facilitator and view the learner’s progress	Progress of the learner is shown to the facilitator
			Login back as the learner and from the group preferences page turn off the progress sharing permissions and login back as the facilitator to view the learner’s progress	Facilitator is not able to view the progress and is shown a message saying that the learner has turned off the progress sharing permissions.
5.	Learner can view their progress through group syllabus	Login as learner who is already part of at least one learner group	From the assigned syllabus section from the learner group page see the current progress and complete a chapter from the the assigned syllabus	No chapter was completed and the progress showed 0%
			Go back to the assigned syllabus section of the learner group page and view he progress now	The progress increases to value not equal to 0%
6.	Learner can exit a learner group they are part of	Login as learner who is already part of at least one learner group	From the learner group page, click on the leave group button	A confirmation modal pops up
			Confirm to leave the learner group	The learner exits the learner group and is redirected to the learner groups dashboard page

Feature testing

Does this feature include non-trivial user-facing changes? **YES**

Implementation Plan

Milestone 1: Facilitators can, from a teacher dashboard page, create a learning group and its syllabus and see all learner groups created by them. They should also be able to view the learner group's homepage (which contains the details of the learning group, its syllabus and the list of learners and their progress).

Table (include both PRs and other actions that need to be taken prior to launch)

No.	Description of PR / action	Prereq PR numbers	Target date for PR creation	Target date for PR to be merged
1	Adding gae_models and domains for learner groups project: ● Adding learner group data and learner group user storage models in core/storage and enlisting them in platform/models.py to make them valid. ● Adding domain files for learner group data and learner group user storage models and writing their test cases.	NA	14th June	21st June
2	Add services for learner groups project: ● Add learner group services and its test file. ● Update user, skill, story services with required functions. ● Add all controllers related to facilitator views in learner_group.py controller and add their respective endpoint redirects in main.py	1	22nd June	29th June
3	Create the learner group backend api service and add required api calls in it. Also create the following learner group models in the frontend:	1, 2	30th June	8th July

	● LearnerGroup model ● LearnerGroupUser model ● LearnerGroupSyllabus model			
4	Add the teacher dashboard page along with the ability to create new learner groups and its syllabus.	1,2,3	8th July	15th July
5	Add the facilitator's view of the learner group page. This should include: ● Ability to view different tabs like homepage, syllabus ● Ability to update learner group info and syllabus ● Ability to delete learner group ● Ability to view all learners that are part of the group. ● Ability to invite new learner to the group by username and remove existing learners	1, 2	15th July	23rd July
	All remaining changes or bug fixes to be done for milestone 1.		19th-24th July	25th July

Milestone 2: Facilitators should be able to invite learners to join the group via username. Learners should be able to join the group, set their preferences (or edit them later), and see the group homepage (which contains their progress, as well as triggers for starting recommended activities from the syllabus).

Table (include both PRs and other actions that need to be taken prior to launch)

No.	**Description of PR / action**	**Prereq PR numbers**	**Target date for PR creation**	**Target date for PR to be merged**
1	Add the common learner groups page showing invitations and learner groups the user is part of.	NA	5th Aug	13th Aug
2	Add learner group joining modal and user being able to join learner group and set	2	13th Aug	21th Aug

	their preferences			
3	Add the learner view of the learner group page. It includes: ● Homepage showing number of skills and stories they have completed ● Progress through the syllabus	NA	20th Aug	29th Aug
	All remaining changes or bug fixes to be done for project completion		27th Aug - 4th Sep	5th Sep

With the current target date , I can make my final submission on the 6th Sep. The final last submission date is 12th Sep. Thus in the worst case I will still have a period of 6-7 days to complete and fix things.

Future Work

- Extend learner groups to have more than one facilitator. Currently we have added backend support for storing multiple facilitators in the learner group, it does not includes:
  - Facilitator being able to add another user as a facilitator.
  - Facilitator being able to remove another facilitator.
  - Handling multiple facilitators making changes to the learner groups simultaneously.

- Currently learners have to remember the common learner groups page url or save it somewhere, we want it to be integrated with the learner dashboard instead in future.

- Currently we only provide an option to add learners via username to the facilitator, later we also want options to add learners via invitation links like discord which would allow learners who do not have an account at oppia to be able to join learner groups.

- The learner group home page does not show any stats related to the learner group in this iteration, moving forward we want it to be integrated with stats services.

- Currently we are not dealing with spamming invitations i.e facilitator sending invitations to learners again and again. It needs to be included later.

- Introduce the concept of user notification based on learner group user flows so that a notification is sent to the affected parties whenever the learner group is updated. It should handle the following flows:
  - A new learner joining the group
  - A learner leaving the group

    - Learner group being deleted
    - A learner being removed from the group
    - Facilitator sending group invitation
    - Facilitator updating the group syllabus