About You

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

I have always been an advocate and a firm believer of education being the only way of bringing positive change in the world. And for this reason, I started working on a project which mapped out to UN's SDG#4 (Quality Education) and came under the Millennium Fellowship by MCN and UNAI in 2020. It worked on the problem of lack of guidance, exposure and direction by providing the means to access information and experiences of others with the aim to bring forth educated decisions regarding career choices.

I wanted to work on something I believed in. The mission of my project coincides with Oppia's, where mine focuses on career guidance, Oppia focuses on learning. and this makes me personally involved and determined to make substantial contributions that I hope could ultimately lead to Oppia's success and in-return help the masses to have access to learning freely. Also, the tech stack used in Oppia is similar to what I have worked on (Python + Angular).

I started working on Angular2+ back in 2020 and had typescript strict checks enabled for all of my projects. I went through quite a learning curve to understand how these are implemented as I wanted to experience using best practices when developing in Angular. Since I have already spent time understanding the fundamentals, I would like to learn more and perfect this knowledge and this project fits perfectly in helping me do just that. It would enable me to find solutions to situations with strict typing that I have not encountered yet.

Prior experience

I have have the following experience with regards to the technical skills needed:

- One year experience using Angular2+
    - Completed Angular Essential Training on Linkedin Learning.
    - The project under the Millennium Fellowship mentioned above uses Angular2+ as the frontend technology.
    - Developed a multi-purpose platform dedicated for the propagation and the use of machine learning and data science using Angular.
- Two years experience using Python
    - Completed Crash Course on Python offered by Google on Coursera
    - The project under the Millennium Fellowship mentioned above uses Django(Python) as the backend technology.
- Two+ years of experience using Github
    - Completed Introduction to Git and GitHub offered by Google on Coursera
- One+ month experience in Quality Assurance
    - Worked as a Software Quality Assurance Trainee at Afiniti.

- Two+ years of Experience in Tech Communities (Non-Technical Skills)
    - Projects Lead at Google's Developer Student Club, NUST
    - Chair Operations at NUST ACM Student Chapter
    - Microsoft Learn Student Ambassador at Microsoft

Links to PRs:

- *#12462 - - Fix Part of #4057: Fully cover NumericInputValidationService with unit tests*
- *#12429 - - Fix Part of #10474: Make TS checks strict for StateEditorRefreshService, StateNameService, StoryNodeModel*
- *#12423 - - Fix Part of #10474 : Make checks strict for SchemaFormSubmittedService and SetInputRulesService*
- *#12411 - - Fix Part of #10474 : Make typescript checks strict for MathEquationInputRulesService*

Contact info and timezone(s)

Name: *Eesha Arif*
Location*: Islamabad, Pakistan*
Contact No*: +92 3134756364*
Education (Ongoing): *Bachelors in Software Engineering*
Institution: *National University of Sciences and Technology, Pakistan*
Primary Email (Hangouts): *eeshaarif@gmail.com*
Secondary Email: *earif.bese18seecs@seecs.edu.pk*
Github: *@EeshaArif*
Linkedin: *https://www.linkedin.com/in/eesha-arif-a9084616b*
Preferred Communication Method: *hangouts, email, slack, discord*
Timezone: *Pakistan Standard Time UTC+05:00*

Time commitment

S.No	Dates (Week)	Days (Total)	Time Commitment
0	17th May - 21st May	Mon - Fri (5)	2h/day - 10h/week
1	24th May - 28th May	Mon - Fri (5)	2h/day - 10h/week
2	31st May - 2nd June	Mon - Wed (3)	2h/day - 6h/week
3	7th June - 12th June	NA	NA
4	15th June - 20th June	Tues - Sun (6)	3h/day - 18h/week

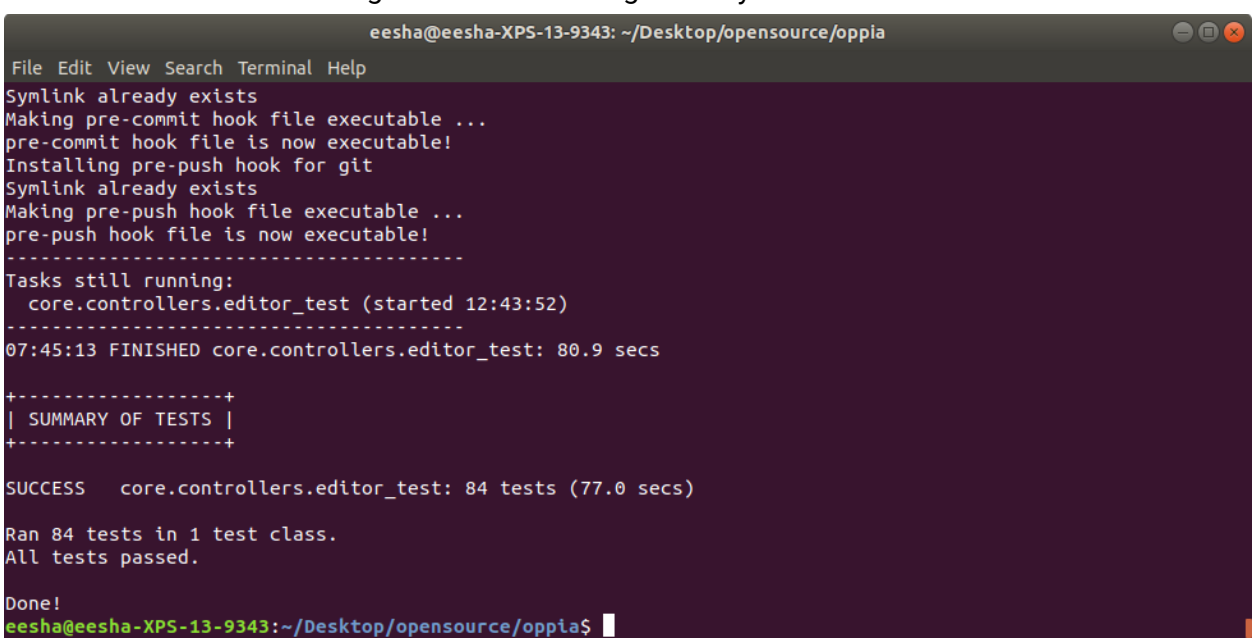
5	21st June - 27th June	Mon - Sun (7)	3h/day - 21h/week
6	28th June - 1st July, 3rd July-4th July	Mon - Thurs, Sat - Sun (6)	3h/day - 18h/week
7	5th July - 9th July	Mon - Fri (5)	3h/day - 15h/week
8	12th July - 16th July	Mon - Fri (5)	3h/day - 15h/week
9	19th July - 25th July	Mon - Fri (7)	3h/day - 21h/week
10	26th July - 1st August	Mon - Sun (7)	3h/day - 21h/week
11	2nd August - 8th August	Mon - Sun (7)	3h/day - 21h/week
12	9th August - 15th August	Mon - Sun (7)	3h/day - 21h/week

Estimated Total Working Days: 68 days
Estimated Total Time Commitment: 197 hours (This can be subject to change according to progress and need)

(The above mentioned estimates were taken to the best of my knowledge at the time of writing this proposal)

Essential Prerequisites

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



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

```
eesha@eesha-XPS-13-9343: ~/Desktop/opensource/oppia
File  Edit  View  Search  Terminal  Help
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4445 of 4447 SUCCESS (0 s
LOG: 'Spec: Interaction attributes extractor service should properly extract mig
rated customization arguments values fromattributes has passed'
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4445 of 4447 SUCCESS (0 s
ecs / 1 min 4.471 secs)
LOG: 'Spec: Interaction attributes extractor service should properly extract mig
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4446 of 4447 SUCCESS (0 s
ERROR: 'Error communicating with server. Please try again.'
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4446 of 4447 SUCCESS (0 s
ecs / 1 min 4.475 secs)
LOG: 'Spec: SvgFilenameEditor initialized with value attribute should load the s
vg file has passed'
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4446 of 4447 SUCCESS (0 s
ecs / 1 min 4.475 secs)
LOG: 'Spec: SvgFilenameEditor initialized with value attribute should load the s
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4447 of 4447 SUCCESS (0 s
Chrome Headless 89.0.4389.114 (Linux x86_64): Executed 4447 of 4447 SUCCESS (1 m
in 15.07 secs / 1 min 4.485 secs)
TOTAL: 4447 SUCCESS
TOTAL: 4447 SUCCESS
09 04 2021 13:13:07.712:WARN [launcher]: ChromeHeadless was not killed in 2000 m
s, sending SIGKILL.
Done!
eesha@eesha-XPS-13-9343:~/Desktop/opensource/oppia$
```

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

```
eesha@eesha-XPS-13-9343: ~/Desktop/opensource/oppia
File  Edit  View  Search  Terminal  Help
[23:32:45] W/element - more than one element found for locator By(css selector,
.protractor-test-learner-dashboard-link) - the first result will be used
[23:32:45] W/element - more than one element found for locator By(css selector,
.protractor-test-learner-dashboard-link) - the first result will be used
.       ? should visit the learner dashboard from the profile dropdown menu
.       ? should not show the topics and skills dashboard link in the profile dro
pdown menu when user is not admin
.       ? should visit the notifications page from the profile dropdown menu
.       ? should visit the preferences page from the profile dropdown menu



8 specs, 0 failures
Finished in 193.197 seconds

Executed 8 of 8 specs SUCCESS in 3 mins 13 secs.
[23:33:47] I/launcher - 0 instance(s) of WebDriver still running
[23:33:47] I/launcher - chrome #01 passed

i  emulators: Received SIGTERM for the first time. Starting a clean shutdown.
i  emulators: Please wait for a clean shutdown or send the SIGTERM signal again
```

Other summer obligations:

I will have classes from 17th May to 2nd June but will still be able to work on the project.

I have final exams for this semester on the 3rd week (7th June - 12th June) and would not be able to work for the specified week (The specific week might change due to Covid).

During writing of this proposal, I have no other summer commitments.

Communication channels

I plan on communicating with my mentor weekly for progress reports and as-needed during the project.
Channels:
I plan on using any one of the following channels:

- Google Meets
- Zoom
- MS Teams
- Hangouts

Application to multiple orgs

I am only applying to Oppia.

Project Details

Product Design

The **users** for this project are the developers on the Oppia team

Oppia uses Angular as the frontend framework with typescript as the primary language, hence this codebase is fully typed which implies that variable assignment, procedure arguments and function return values will all be associated explicitly with a type.

Moreover, these type checks are enforced during compile time which signifies that exceptions and errors are more likely to occur during compilation. Hence, this explicit typing makes code self-documenting, produces less bugs, enhances understanding of the code and reduces wastage of time from debugging errors at runtime.

`**strict**` is a typescript compiler option which turns on the following set of rules (**strict mode**):

- noImplicitAny
    - Variables/function arguments cannot have implicit type `any`
- noImplicitThis
    - The context of `this` cannot be defined implicitly
- strictNullChecks
    - Values can be null or undefined only if explicitly marked
- strictPropertyInitialization
    - All class properties need to be initialized in a constructor or property initializer
- strictBindCallApply
    - Enforces stricter checking of `bind`, `call` and `apply` functions
- strictFunctionTypes
    - Argument types cannot be bivariant

Enabling the above rules helps to reduce the chances of getting unpredictable results and makes the code more robust but currently, Oppia's code base does not pass these strict rules. This makes the code prone to unexpected actions and errors.

To avoid this, the following actions need to be taken:

1. All the new files to be added should have typescript strict mode enabled
2. Strict typing should be introduced to the files already present in the code base

In this project, I will change the typescript config strict file to ensure that all newly added files need to pass these strict rules. After this, I will take 120 files and their tests (120 + 120 = 240) already present in the code base and enforce these strict typing checks on them. These files will be chosen from **UpgradedServices.ts** which has files listed down in eleven topological levels (0-10). The files will be updated in ascending order of topological level, hence the files listed at topological level 0 will be chosen first and then the ones present at a higher level will be considered.

Upon the completion of the project, any new file added to the Oppia's code base will, by default, have typescript **strict mode** enabled and hence would need to pass all the above mentioned rules. Also, strict typing will be introduced to some 120 files listed down in **UpgradedServices.ts** alongside their test files (240 in total).

Technical Design

Architectural Overview

The Oppia codebase currently has two typescript config files; *tsconfig.json* and *tsconfig-strict.json*.

tsconfig-strict.json:

tsconfig-strict.json compiler options has ***strict*** set to ***true*** which enables the five rules namely *noImplicitAny*, *noImplicitThis*, *strictNullChecks*, *strictPropertyInitialization*. *strictBindCallApply* and *strictFunctionTypes*.
It currently lists down specific files which pass the strict typescript checks. This list has been updated incrementally after introducing the strict rules to individual files since all the files cannot be covered at once as the compiler throws a significant number of errors.

These config options (strict mode), hence, are only enabled for the files listed down in *tsconfig-strict.json*.

tsconfig.json:

Whereas in ***tsconfig.json*,** these strict rules are disabled by setting ***noImplicitUseStrict*** as ***true*** in the compiler options. This config file currently includes all the groups of files present in Oppia codebase folders "*core*", "*extensions*" and "*typings"*. Hence, the strict mode is disabled for all the files in the folders when this configuration is used.

typescript_checks.py:

typescript_checks.py is the script used for compiling and checking typescript. It compiles the files using the configuration stated in a typescript config file. If an optional flag "***--strict_checks***" is added then it compiles the files using *tsconfig-strict.json.* If the flag is removed then the *tsconfig.json* file is used instead.

No	Command	Config File
1	python -m scripts.typescript_checks	tsconfig.json
2	python -m scripts.typescript_checks --strict_checks	tsconfig-strict.json

config.yml:

Oppia uses CircleCI for continuous integrations. The CircleCI tests run the following job.

```yaml
jobs:
  setup_and_typescript_tests:
    <<: *job_defaults
    steps:
      - checkout
      - merge_target_branch
      - run:
          name: Setup python by installing wheel
          command: pip install wheel==0.35.0
      - run:
          name: Install dependencies
          command: python -m scripts.install_third_party_libs
      - run:
          name: Check that all e2e test files are captured in protractor.conf.js
          command: python -m scripts.check_e2e_tests_are_captured_in_ci
      - run:
          name: Run typescript tests
          command: |
            python -m scripts.typescript_checks
      - run:
          name: Run typescript tests in strict mode
          command: |
            python -m scripts.typescript_checks --strict_checks
      - persist_to_workspace:
          root: /home/circleci/
```

It runs *python -m scripts.typescript_checks:*

- All the files in the folder paths included in tsconfig.json will be compiled and checked without strict rules
- The folders included were *core, extensions,* and *typings*

Then it runs *python -m scripts.typescript_checks --strict_checks*

- The specific files included in tsconfig-strict.json will be compiled again but checked with strict rules
- These specific files were also present in "*core", "extensions"* and/or "*typings"* folder.

pre_push_hook.py:

This hook also performs the same action as above. It runs both the commands and checks for errors before pushing to the repository.

```
typescript_checks_status = 0
if does_diff_include_ts_files(files_to_lint):
    typescript_checks_status = run_script_and_get_returncode(
        TYPESCRIPT_CHECKS_CMDS)
if typescript_checks_status != 0:
    python_utils.PRINT(
        'Push aborted due to failing typescript checks.')
    sys.exit(1)

strict_typescript_checks_status = 0
if does_diff_include_ts_files(files_to_lint):
    strict_typescript_checks_status = run_script_and_get_returncode(
        STRICT_TYPESCRIPT_CHECKS_CMDS)
if strict_typescript_checks_status != 0:
    python_utils.PRINT(
        'Push aborted due to failing typescript checks in '
        'strict mode.')
```

Implementation Approach

The following tasks need to be performed:

1. Make all newly added files enforce TS strict checks
2. Make TS checks strict for files already present in the codebase.
    - 2.1. Choose the files to cover and their order
    - 2.2. Introduce strict typescript checks to chosen files

1. Make all newly added files enforce TS strict checks

The *tsconfig-strict.json* file needs to be changed to make all newly added typescript files compile with strict mode enabled.

Currently it has a ***files*** property which is an array which lists down all the files that will be included with the configuration set to strict mode.

The tsc CLI does not provide any flag for just outputting the name of files in which error occurs.

We need to get a list of files which do not pass strict checks and exclude them, the following steps will be taken:

1. The ***include*** property will be updated to include *core* and *extensions* folder in its paths.

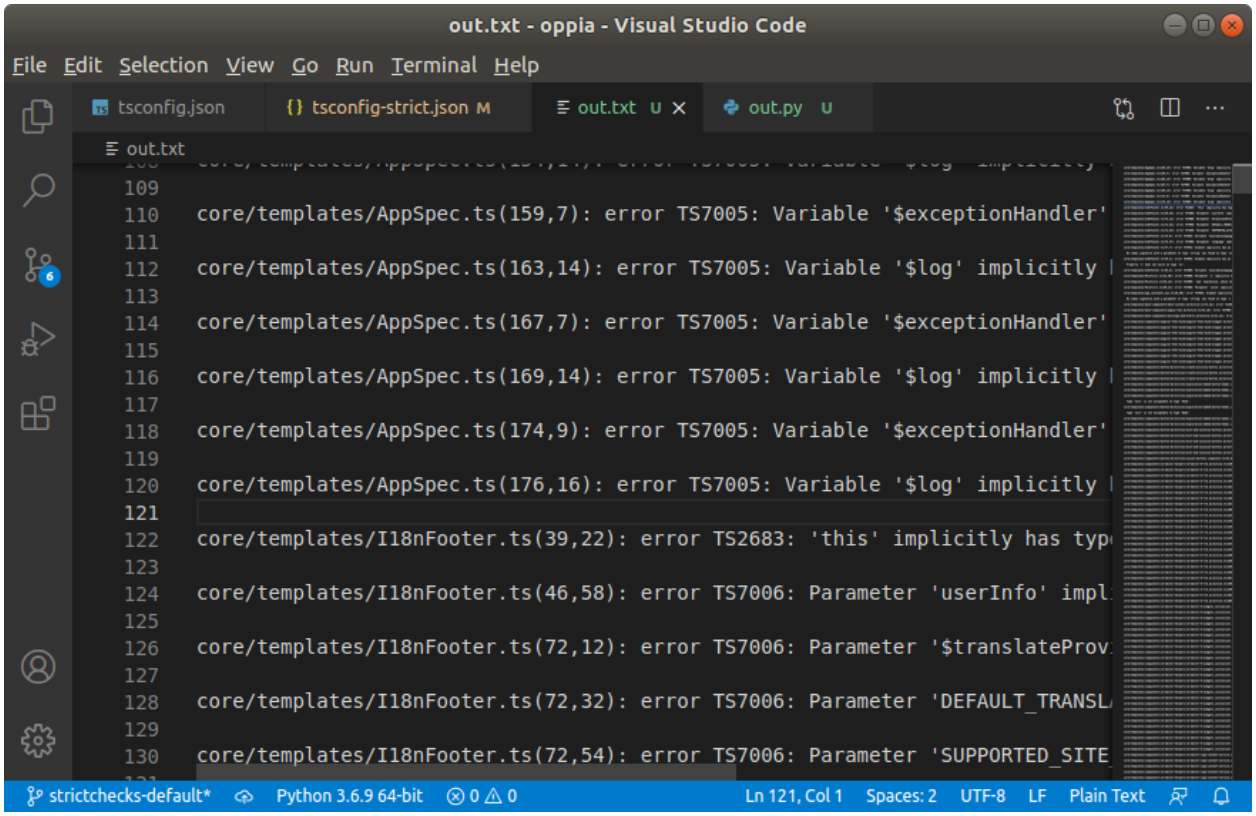
```
"include": ["core/*.ts","extensions/*.ts","typings/*.ts"]
```

2. The following command will be run

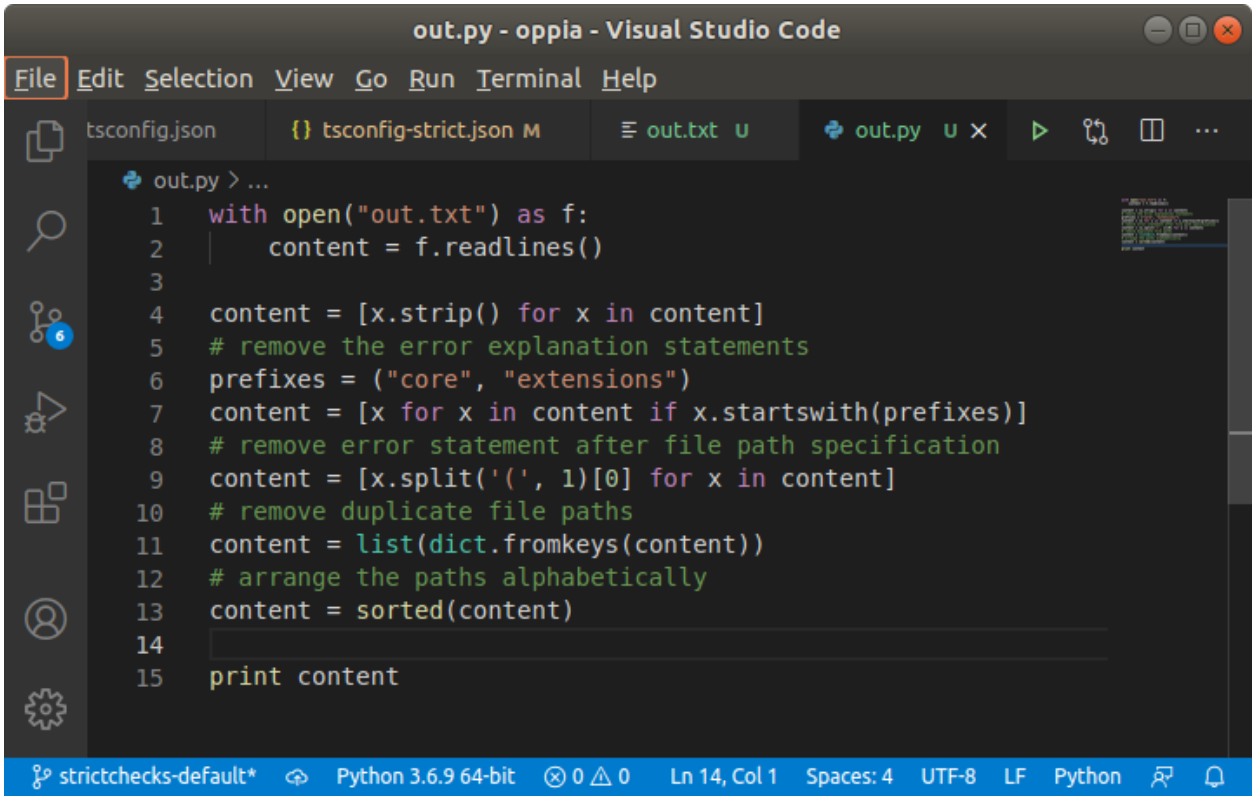
```
python -m scripts.typescript_checks --strict_checks > out.txt
```

This will log the error output in a text file named *out.txt*

3. The text file looks like this



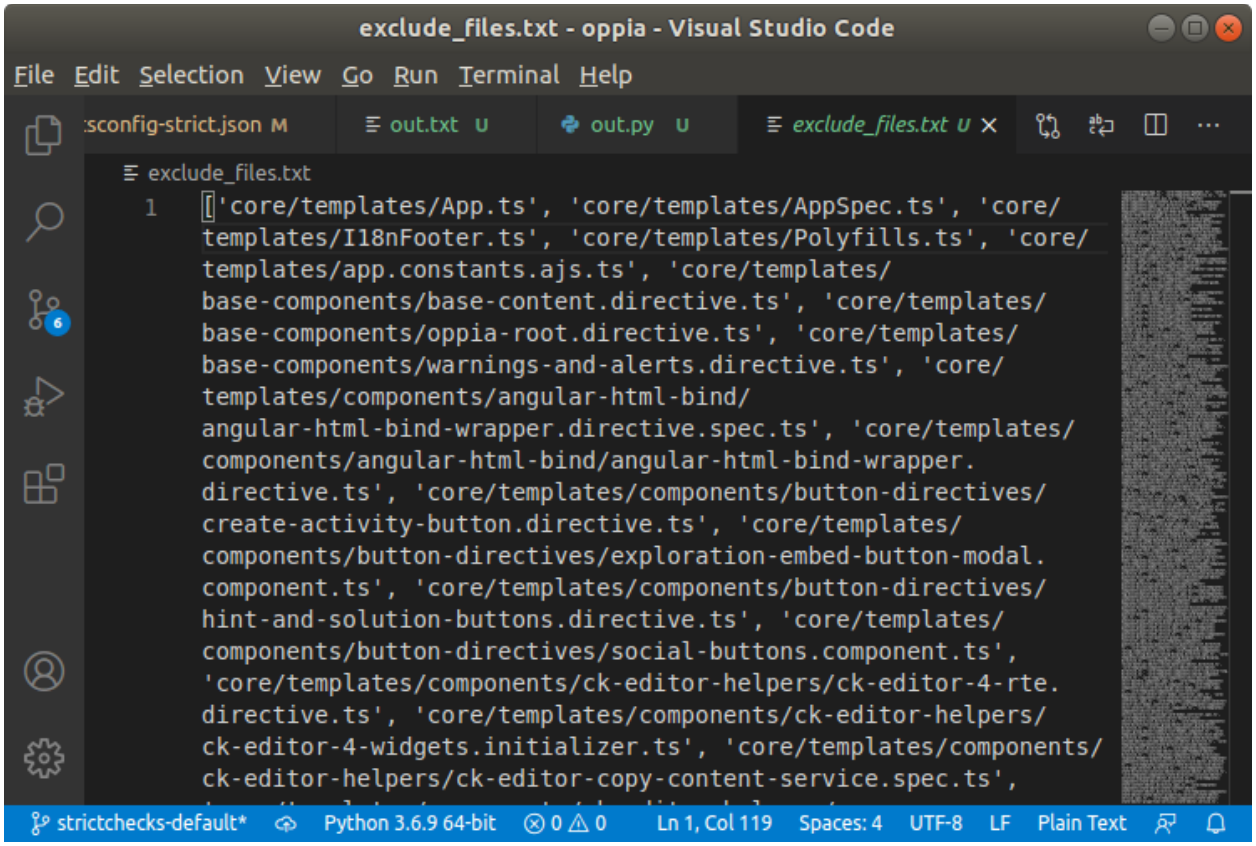
4. The following is a python script that will clean the output



5. And then by running the following command, we will get the **list of all the files** in the Oppia codebase that currently **do not pass strict checks** in the text file named ***exclude_files.txt***.

```
python out.py > exclude_files.txt
```

The file looks like this



6. We add another property ***exclude*** in *tsconfig-strict.json* and set its value to the list of file paths in ***exclude_files.txt.***
7. And lastly, we remove the ***files*** property alongside its file paths since they are already being covered in ***include.***

These steps will enable strict mode for every other new file that will be added in the folders *'core', 'extensions' and 'typings'* but will exclude the files already present in the codebase that do not pass these checks.

2. Make TS checks strict for files already present in the codebase

2.1 Choose the files to cover and their order:

The Oppia Foundation had already started introducing strict typing to files present at the **topological level 0** in UpgradedServices.ts. And this is being tracked with #10474: Make typescript checks strict.

As stated before, the files will be updated in ascending order of topological level, hence the files listed at topological level 0 will be chosen first and then the ones present at a higher level will be considered but the **order can change** if a file present at a lower level has a dependency of a file present at a higher level. The dependency will be catered first.

On further inspection of this issue together with files listed down in tsconfig-strict.json and UpgradedServices.ts, the state of the files during writing of this proposal is as follows:

Assigned Files: (Not Covered):
These files are already assigned to contributors and will only be covered during the project if a dependency exists with another file.

1. baseInteractionValidationService
2. CollectionValidationService
3. ComputeGraphService
4. ExplorationFeaturesService
5. ImprovementsService
6. InteractionSpecsService
7. MusicPhrasePlayerService
8. NumericExpressionInputRulesService

Topological Level 0:
Total: 90 files
Not Covered: 32 files

No	File	Dependencies
1	ExplorationDiffService **(No Test File)**	InteractionObjectFactory, RecordedVoiceOverObjectFactory, NumberWithUnitsObjectFactory, ExtensionTagAssemblerService, StatesObjectFactory, StateObjectFactory, ParamTypeObjectFactory, RuleObjectFactory, UnitsObjectFactory, WrittenTranslationObjectFactory
2	GraphUtilsService	InteractionObjectFactory, NumberWithUnitsObjectFactory, RecordedVoiceOverObjectFactory, ExtensionTagAssemblerService, StatesObjectFactory, StateObjectFactory, ComputeGraphService, RuleObjectFactory.ts, StateGraphLayoutService, UnitsObjectFactory, WrittenTranslationObjectFactory

3	LearnerActionObjectFactory	NA
4	LostChangeObjectFactory	NumberWithUnitsObjectFactory, RecordedVoiceOverObjectFactory, ExtensionTagAssemblerService, StatesObjectFactory, StateObjectFactory, ComputeGraphService, ParamTypeObjectFactory.ts, RuleObjectFactory.ts, UnitsObjectFactory, UtilsService, WrittenTranslationObjectFactory
5	MisconceptionObjectFactory	NA
6	NumberAttemptsService	NA
7	ParamTypeObjectFactory	NA
8	PlayerCorrectnessFeedbackEnabledService **(No Test File)**	NA
9	PlaythroughIssueObjectFactory	NA
10	RatingComputationService	NA
11	RatioExpressionInputRulesService	RatioModel
12	ReviewTestEngineService	NA
13	RubricObjectFactory	NA
14	RuleObjectFactory	NA
15	ShortSkillSummaryObjectFactory	NA
16	SolutionValidityService	NA
17	StateGraphLayoutService **(No Test File)**	InteractionObjectFactory, RecordedVoiceOverObjectFactory, NumberWithUnitsObjectFactory, ExtensionTagAssemblerService, StatesObjectFactory, StateObjectFactory, ComputeGraphService, RuleObjectFactory.ts, StateGraphLayoutService, UnitsObjectFactory,

		WrittenTranslationsObjectFactory
18	StoryContentsObjectFactory	NA
19	StoryObjectFactory	StoryContentsObjectFactory
20	StoryReferenceObjectFactory	NA
21	SuggestionModalService	NA
22	SuggestionsService	NA
23	TextInputTokenizer	NA
24	ThreadStatusDisplayService	NA
25	TopicsAndSkillsDashboardPageService	NA
26	UnitsObjectFactory	NA
27	UtilsService	NA
28	VersionTreeService	InteractionObjectFactory, RecordedVoiceOverObjectFactory, NumberWithUnitsObjectFactory, ExtensionTagAssemblerService, ParamTypeObjectFactory, RuleObjectFactory, UnitsObjectFactory, WrittenTranslationObjectFactory
29	VoiceoverObjectFactory	NA
30	WindowRef	NA
31	WinnowingPreprocessingService	NA
32	WrittenTranslationObjectFactory	NA

Topological Level 1:

No	File	Dependencies
1	AlgebraicExpressionInputValidationService	NA
2	AlertsService	NA
3	BrowserCheckerService	NA

4	CodeReplValidationService	RuleObjectFactory
5	ContinueValidationService	RuleObjectFactory
6	DeviceInfoService	NA
7	DocumentAttributeCustomizationService	NA
8	DragAndDropSortInputValidationService	RuleObjectFactory
9	ExpressionSyntaxTreeService	NA
10	FeedbackThreadObjectFactory	ThreadMessageModel
11	FractionInputRulesService	UtilsService
12	FractionInputValidationService	RuleObjectFactory
13	GraphInputRulesService	GraphUtilsService, UtilsService
14	GraphInputValidationService	RuleObjectFactory
15	GuestCollectionProgressService	CollectionNodeModel, CollectionProgressModel
16	HintObjectFactory	NA
17	HtmlEscaperService	NA
18	ImageClickInputValidationService	RuleObjectFactory
19	InteractiveMapValidationService	RuleObjectFactory
20	ItemSelectionInputValidationService	RuleObjectFactory
21	LocalStorageService	InteractionObjectFactory, ParamTypeObjectFactory, RecordedVoiceOverObjectFactory, RuleObjectFactory, WrittenTranslationObjectFactory, NumberWithUnitsObjectFactory, UnitsObjectFactory, ExtensionTagAssemblerService
22	MathEquationInputValidationService	RuleObjectFactory
23	MessengerService **(No Test File)**	NA
24	MetaTagCustomizationService	NA

25	MultipleChoiceInputValidationService	RuleObjectFactory
26	MusicNotesInputRulesService	UtilsService
27	MusicNotesInputValidationService	NA
28	NormalizeWhitespacePipe	UtilsService
29	NumericInputValidationService	RuleObjectFactory
30	NumberWithUnitsObjectFactory	UnitsObjectFactory
31	NumericExpressionInputValidationService	RuleObjectFactory
32	NumberWithUnitsRulesService	UnitsObjectFactory, UtilsService. NumberWithUnitsObjectFactory
33	OutcomeObjectFactory	NA
34	PageTitleService	NA
35	ParamChangesObjectFactory	NA
36	ParamSpecObjectFactory	ParamTypeObjectFactory
37	ParamTypeObjectFactory	NA
38	PencilCodeEditorValidationService	RuleObjectFactory
39	PlayerTranscriptService	ExtensionTagAssemblerService, AudioTranslationLanguageService, LanguageUtilService, StateCardObjectFactory, UnitsObjectFactory, NumberWithUnitsObjectFactory, WrittenTranslationsObjectFactory, RuleObjectFactory, RecordedVoiceoverObjectFactory, InteractionObjectFactory
40	PlaythroughObjectFactory	NA
41	PythonProgramTokenizer	NA
42	QuestionValidationService	ExtensionTagAssembler, SolutionValidityService, StateObjectFactory, QuestionObjectFactory, UnitsObjectFactory,

		NumberWithUnitsObjectFactory, WrittenTranslationsObjectFactory, RuleObjectFactory, RecordedVoiceOverObjectFactory, InteractionObjectFactory
43	RatioExpressionInputValidationService	RatioModel, RuleObjectFactory
44	RecordedVoiceoversObjectFactory	NA
45	SetInputValidationService	RuleObjectFactory
46	SkillCreationBackendApiService	NA
47	StateTopAnswersStatsObjectFactory	NA
48	SpeechSynthesisChunkerService	NA
49	SVMPredictionService	NA
50	SchemaDefaultValueService	NA
51	SiteAnalyticsService	NA
52	StateClassifierMappingService	NA
53	StateEditorService	ExtensionTagAssemblerService, SolutionValidityService, StateObjectFactory, QuestionObjectFactory, UnitsObjectFactory, NumberWithUnitsObjectFactory, WrittenTranslationsObjectFactory, RuleObjectFactory, RecordedVoiceOverObjectFactory, InteractionObjectFactory
54	StoryContentsObjectFactory	NA
55	SubtopicObjectFactory	NA
56	TextInputValidationService	TextInputRulesService, UtilsService, RuleObjectFactory
57	TopicCreationBackendApiService	NA
58	UrlService	NA
59	WindowDimensionsService	NA

60	WorkedExampleObjectFactory	NA
61	WrittenTranslationsObjectFactory	NA

Topological Level 2:

No	File	Dependencies
1	AnswerGroupObjectFactory **(No Test File)**	RuleObjectFactory
2	CkEditorCopyContentService	NA
3	AutogeneratedAudioPlayerService **(No Test File)**	SpeechSynthesisChunkerService
4	BottomNavbarStatusService	PreventPageUnloadEventService
5	PreventPageUnloadEventService	NA
6	CodeReplPredictionService	WinnowingPreprocessingService. SVMPredictionService, PythonProgramTokenizer
7	CodeReplRulesService	UtilsService
8	ConceptCardObjectFactory	RecordedVoiceOverObjectFactory
9	EditorFirstTimeEventsService	NA
10	ExpressionSyntaxTreeService	NA
11	ExtensionTagAssemblerService	NA
12	FocusManagerService	NA
13	NumberWithUnitsValidationService	UnitsObjectFactory, NumberWithUnitsObjectFactory, RuleObjectFactory
14	ParamSpecsObjectFactory	UtilsService
15	SidebarStatusService	NA
16	SubtopicPageContentsObjectFactory	RecordedVoiceOverObjectFactory
17	TopicObjectFactory	SubtopicObjectFactory

18	ValidatorsService	UtilsService

Some files will also be covered which are not present in UpgradedServices.ts but are a dependency.

The order in which the strict checks will be introduced to the files is listed down in Milestones.

2.2 Introduce strict typescript checks to chosen files:

When converting the current Oppia's codebase files to strict mode, the following are **some** of the recurring situations which arise alongside the solutions on how to solve them.

Situation 1:

The type of variable is not defined explicitly and hence typescript assigns it an '*any*' type. This violates the ***noImplicitAny*** rule of strict mode. This situation mostly occurs in the test files of the Oppia codebase.

```
import {
  'class
                (local var) service: any
                Variable 'service' implicitly has type 'any' in some locations
describe        where its type cannot be determined. ts(7034)
  descri        View Problem (Alt+F8)    Quick Fix... (Ctrl+.)
    var service;
    beforeEach(() => {
      service = new WinnowingPreprocessingService();
    });
```

To solve this, the variable is explicitly assigned the type it belongs to. In some cases in the Oppia codebase, we may need to import the Interface, Class or Type as it is not imported in the file before.

In the example, the service was of type *WinnowingPreprocessingService*.

```
describe('Winnowing preprocessing functions', () => {
  describe('Test winnowing preprocessing functions', () => {
    var service: WinnowingPreprocessingService;
    beforeEach(() => {
      service = new WinnowingPreprocessingService();
    });
```

Situation 2:

The ***strictNullChecks*** rule disallows assigning *null* and *undefined* as a value until the type is explicitly marked.
In the following example taken from the codebase, *ruleObjectFactory* is of type *RuleObjectFactory.* Since it was not explicitly assigned the type *null,* the compiler throws an error when we try to assign it the value *null.*

```
import  let ruleObjectFactory: RuleObjectFactory
  'dom
        Type 'null' is not assignable to type 'RuleObjectFactory'. ts(2322)
descri  View Problem (Alt+F8)   No quick fixes available
  let ruleObjectFactory: RuleObjectFactory = null;
  let ruleBackendDict: RuleBackendDict = null;
  let inputBackend: RuleInputs = null;
```

In some cases that arise throughout the Oppia codebase as in this example, we can simply remove the *null* value assignment.

```
import { RuleObjectFactory, RuleBackendDict, RuleInputs, Rule } from
  'domain/exploration/RuleObjectFactory';

describe('RuleObjectFactory', () => {
  let ruleObjectFactory: RuleObjectFactory;
  let ruleBackendDict: RuleBackendDict;
  let inputBackend: RuleInputs;
```

Situation 3:

In cases where the ***strictNullChecks*** rule does not pass and the value *null* or *undefined* is being passed to a function, the first course of action to be taken is to try and refactor the code in a way that adding *null* or *undefined* may not be needed.

```
  provided
})
export cla
  private
  private savedMemento: string = null;
  /**
   * @param {string} value - Memento of the state name to be set to.
   */
   setStateNameSavedMemento(stateName: string): void {
    this.savedMemento = stateName;
  }
```

```
(property) StateNameService.savedMemento: string
Type 'null' is not assignable to type 'string'. ts(2322)
View Problem (Alt+F8)    No quick fixes available
```

We can try to refactor it by assigning it an empty string **but** the initial value of *savedMemento* has to be *null.*

The value *null* for savedMemento is being passed here;

```
    return this.savedMemento;
  }

  init(): void {
    this.setStateNameSavedMemento(null);
    this.setStateNameEditorVisibility(false);
  }
}
```

```
Argument of type 'null' is not assignable to
parameter of type 'string'. ts(2345)
View Problem (Alt+F8)    No quick fixes available
```

If refactoring may not be an option as in the example above, we can always add the types *null* explicitly.

```
  private savedMemento: string | null = null;
  /**
   * @param {string} value - Memento of the state name to be set to.
   */
   setStateNameSavedMemento(stateName: string | null): void {
    this.savedMemento = stateName;
  }
```

Situation 4:

In situations where the type is *undefined* and cannot be refactored.

```
(local var) height: number | undefined
Object is possibly 'undefined'. ts(2532)
View Problem (Alt+F8)    No quick fixes available
var heig
return (height > 630);
```

height() will only return undefined if shadowPreviewCard Selector does not exist.
The css classes do exist in the directives html code and was put there for the exact purpose of checking the height of the card and hence the selector will be valid and the undefined case will never occur as of this state of the code.

```
<!--
  Off-screen preview of the learner view card in order to check if the height of the
  card has exceeded a specified limit.
-->
<div class="oppia-shadow-preview-card" aria-hidden="true">
  <div class="oppia-learner-view-card-content">
    <div class="oppia-learner-view-card-top-section">
      <angular-html-bind class="oppia-rte-viewer oppia-learner-view-card-top-content"
                         html-data="StateContentService.displayed._html">
      </angular-html-bind>
    </div>
  </div>
</div>
```

We could just use the non-null assertion operator to assert that the undefined case will never occur but with the @typescript-eslint/no-non-null-assertion enabled, the non-null assertion operator is forbidden.

```
$scope.isCardHeightLimitReached = function() {
  var shadowPreviewCard = $(
    '.oppia-shadow-preview-card .oppia-learner-view-card-top-section'
  );
  var height = shadowPreviewCard.height()!;
  return (height > 630);
};
```

And there could also be a situation where the code is updated and accidently the wrong classes or selectors are used. In that case, we should not send wrong height warnings to the user.

```
$scope.isCardHeightLimitReached = function() {
  // TODO Add A Test Case for invalid selector
  var shadowPreviewCard = $(
    '.oppia-shadow-preview-card .oppia-learner-view-card-top-section'
  );
  var height = shadowPreviewCard.height();
  return (height? height > 630: false);
};
```

Note: (undefined > 630) would have returned false even if not stated explicitly above but explicitly stating it makes it easier to debug if the need arises.

A **test case** should be added in its spec file for the scenario where the wrong selector may have been used to make sure the error is caught before any code change is accepted.
Another approach to be used is to simply throw an error.

```
$scope.isCardHeightLimitReached = function() {
  var shadowPreviewCard = $(
    '.oppia-shadow-preview-card .oppia-learner-view-card-top-section'
  );
  var height = shadowPreviewCard.height();
  if (typeof height === undefined){
    throw new Error("Shadow Preview Card Selector does not exist");
  }

```

Situation 5:

Using *?* before assigning values to properties makes them optional, which means that they can have type *undefined* alongside whichever type they were assigned.

For example, *can_edit_topic* is implicitly of type *boolean* | *undefined.*

```
  // These properties are optional because they are only present in the
  // topic summary dict of topic dashboard page.
  'can_edit_topic'?: boolean;
  'is_published'?: boolean;
  'classroom'?: string;
  'url_fragment': string;
}
```

A case can arise where these properties are sure to be assigned a value (they are surely non-undefined) and are used as an argument (or any other assignment) which is not of type *undefined.* At times the *strictNullChecks* will not allow this since the compiler (type checker) fails to conclude the fact that the value assigned to these properties will surely never be *undefined* and throws an error.

```
(parameter) topicSummaryBackendDict: TopicSummaryBackendDict

Argument of type 'boolean | undefined' is not assignable to
parameter of type 'boolean'.
  Type 'undefined' is not assignable to type 'boolean'. ts(2345)

View Problem (Alt+F8)    No quick fixes available
topicSummaryBackendDict.can_edit_topic,
topicSummaryBackendDict.is_published,
topicSummaryBackendDict.classroom,
topicSummaryBackendDict.thumbnail_filename,
topicSummaryBackendDict.thumbnail_bg_color,
topicSummaryBackendDict.url_fragment);
}
```

A solution to this is to use the **Non-null assertion operator (!)**, it asserts the type checker that its operand will be *non-undefined* and *non-null* or in other words, excludes the type *undefined* and *null* from the operand's type.

```
topicSummaryBackendDict.topic_model_last_updated,
topicSummaryBackendDict.can_edit_topic!,
topicSummaryBackendDict.is_published!,
topicSummaryBackendDict.classroom!,
topicSummaryBackendDict.thumbnail_filename,
topicSummaryBackendDict.thumbnail_bg_color,
topicSummaryBackendDict.url_fragment);
}
```

Situation 6:

The ***strictPropoertyInitialization*** rule enforces that the properties be assigned either in a constructor or with a property initializer and due to this, the following situation may occur.

```
styleUrls
})
export clas
(property) SkillMasteryViewerComponent.masteryChange: number

Property 'masteryChange' has no initializer and is not definitely assigned in
the constructor. ts(2564)

View Problem (Alt+F8)    Quick Fix... (Ctrl+.)
@Input()
@Input() masteryChange: number;
```

We can initialize them in the constructor but there are different number of ways to solve this:

We can use the assertion operator if we are sure the value will be assigned at runtime before being used.

```
@Input() skillId!: string;
@Input() masteryChange!: number;
```

If that is not the case, we can make it optional and put a check for edge cases throughout the code where the property might be *undefined*.

```
@Input() skillId?: string;
@Input() masteryChange?: number;
```

Situation 7:

This situation or a variant of it arises in a significant number of files in the Oppia codebase when working with dictionaries. The following error arises because there is no explicit type mentioned for the key-value pairs of the dictionary. Hence the type of value returned for the specific key cannot be determined and ends up implicitly with type '*any'* which is not allowed in typescript strict mode.

In the example, *nodeTitles* had type *{ }* with no explicit mentioning of the types of key-value pairs and *nodes[ ].getId()* returned a *string.*

```
}

getNod
  var
  var
  for
    if
      nodeTitles[nodes[i].getId()] = nodes[i].getTitle();
    }
  }
```

```
(local var) nodes: StoryNode[]

Element implicitly has an 'any' type because expression of type
'string' can't be used to index type '{}'.
  No index signature with a parameter of type 'string' was found
on type '{}'. ts(7053)

View Problem (Alt+F8)    No quick fixes available
```

To solve this, we need to explicitly mention the type of key-value pair/s present in the dictionary which is done in the following example in the second statement of the function.

```
getNodeIdsToTitleMap(nodeIds: string[]): {} {
  var nodes = this._nodes;
  var nodeTitles: {[key:string]: string} = {};
  for (var i = 0; i < nodes.length; i++) {
    if (nodeIds.indexOf(nodes[i].getId()) !== -1) {
      nodeTitles[nodes[i].getId()] = nodes[i].getTitle();
    }
  }
```

Another approach that could be taken is to define an interface.

```
}
interface NodeTitles {
  [node_id: string] : string
}
```

And that interface type declaration can be used instead.

```
getNodeIdsToTitleMap(nodeIds: string[]): {} {
  var nodes = this._nodes;
  var nodeTitles: NodeTitles = {};
  for (var i = 0; i < nodes.length; i++) {
    if (nodeIds.indexOf(nodes[i].getId()) !== -1) {
      nodeTitles[nodes[i].getId()] = nodes[i].getTitle();
    }
  }
```

Null refactor:

We should always change the spec file to adhere to the actual file and hence for the following null assertions, it is better to refactor the code

```
Input > directives > TS music-notes-input-validation.service.spec.ts > describe('MusicNotesInputValidationService') callback
47        feedback: {
48          html: '',
49          content_id: ''
50        },
51        labelled_as_correct: false,
52        param_changes: [],
53        refresher_exploration_id: null,
54        missing_prerequisite_skill_id: null
55      });
56      goodAnswerGroups = [agof.createNew([], goodDefaultOutcome, null, null
57    });
58
```

```
Argument of type 'null' is not assignable to parameter of type 'readonly InteractionAnswer[]'. ts(2345)
View Problem (Alt+F8)        No quick fixes available
```

Which can be done as follows:

```
55    });
56    goodAnswerGroups = [agof.createNew([], goodDefaultOutcome, [], '')];
57  );
58
```

The below case demonstrates refactoring the code not in the test file

```
39  })
40  ex
41
42
43
44
45    currentAudioLanguageCode: string = null;
46    allAudioLanguageCodesInExploration: string[] = null;
47    explorationLanguageCode: string = null;
48    automaticTextToSpeechEnabled: boolean = null;
49    languagesInExploration: ExplorationLanguageInfo[] = [];
50
51    attemptToSetAudioLanguageToExplorationLanguage(): void {

(property)
AudioTranslationLanguageService._currentAudioLanguageCode:
string

Type 'null' is not assignable to type 'string'. ts(2322)

View Problem (Alt+F8)   No quick fixes available

null-strict*  Python 3.6.9 64-bit  25 0  Spaces: 2  UTF-8  LF  TypeScript  4.2.3  Prettier
```

We can assign them the following initial values

```
44
45    _currentAudioLanguageCode: string = '';
46    _allAudioLanguageCodesInExploration: string[] = [];
47    _explorationLanguageCode: string = '';
48    _automaticTextToSpeechEnabled: boolean = false;
49    _languagesInExploration: ExplorationLanguageInfo[] = [];
50
```

Situation 8:

The following demonstrates how to handle constants.
If the following scenario arises where we need to specify the type

```
ribe('NumberWithUnitsObjectFactory', () => {
scribe('number with units
var nwuof: NumberWithUnit
var uof: UnitsObjectFacto
var errors;

beforeEach(() => {
  nwuof = new NumberWithU
    new UnitsObjectFactor
  uof = new UnitsObjectFa
  errors = ObjectsDomainConstants.NUMBER_WITH_UNITS_PARSING_ERRORS;
});

(property) NUMBER_WITH_UNITS_PARSING_ERRORS: {
    readonly INVALID_VALUE: "Please ensure that value is
either a fraction or a number";
    readonly INVALID_CURRENCY: "Please enter a valid
currency (e.g., $5 or Rs 5)";
    readonly INVALID_CURRENCY_FORMAT: "Please write
currency units at the beginning";
    readonly INVALID_UNIT_CHARS: string;
}
```

then its solution is to use typeof <constant> at line 32

```
32    var errors: typeof ObjectsDomainConstants.NUMBER_WITH_UNITS_PARSING_ERRORS;
33
34    beforeEach(() => {
35      nwuof = new NumberWithUnitsObjectFactory(
36        new UnitsObjectFactory(), new FractionObjectFactory());
37      uof = new UnitsObjectFactory();
38      errors = ObjectsDomainConstants.NUMBER_WITH_UNITS_PARSING_ERRORS;
39    });
```

Situation 9:

When it comes to constants and their keys, the following situation may arise

```
193     var keys = Object.keys(ObjectsDomainConstants.CURRENCY_UNITS);
194
195       for (var i = 0; i < keys.length; i++) {
196         for (
197           var j = 0;
198           j <
199           j++)
200           if (
201             un
202
203             un
204
205             un
206           }
207         }
208
209         for (
210           var
211           j < ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases.length;
212           j++) {
213           if (
214             units.includes(
215               ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases[j])) {
216             units = units.replace(
217               ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases[j],
218               ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].name);
219           }
```

```
import ObjectsDomainConstants

Element implicitly has an 'any' type because expression of type
'string' can't be used to index type '{ readonly dollar: {
readonly name: "dollar"; readonly aliases: readonly ["$",
"dollars", "Dollars", "Dollar", "USD"]; readonly front_units:
readonly ["$"]; readonly base_unit: null; }; readonly rupee: {
readonly name: "rupee"; readonly aliases: readonly [...]; readonly
front_units: readonly [...]; readonly base_uni...'.
  No index signature with a parameter of type 'string' was found
on type '{ readonly dollar: { readonly name: "dollar"; readonly
aliases: readonly ["$", "dollars", "Dollars", "Dollar", "USD"];
readonly front_units: readonly ["$"]; readonly base_unit: null; };
readonly rupee: { readonly name: "rupee"; readonly aliases:
```

The potential fix to this error is to explicitly identify the type of key values to be of those properties present in the constant.

```
var keys = <Array<keyof typeof ObjectsDomainConstants.CURRENCY_UNITS>>Object.keys(
  ObjectsDomainConstants.CURRENCY_UNITS
  );

for (var i = 0; i < keys.length; i++) {
  for (
    var j = 0;
    j < ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].front_units.length;
    j++) {
    if (
      units.includes(
        ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].front_units[j])) {
```

Situation 10:

Another situation that arose while dealing with constants was the following:
createUnit is a function that belongs to math.js library and hence its function definition could not be changed.
The function expects the following parameters

Parameter	Type
`name`	string
`definition`	string, Unit
`options`	Object

```
(alias) createUnit(name: string, definition?: string |
math.UnitDefinition | undefined, options?: math.CreateUnitOptions
| undefined): math.Unit (+1 overload)
```

but the CURRENCY_UNITS properties are of type readonly and direct conversion to type string[] is not possible.

```
["cents", "cents", "cent"] | readonly ["paisa", "Paise", "Paisa"]'
is not assignable to type 'string[] | undefined'.
      The type 'readonly ["$", "dollars", "Dollars", "Dollar",
"USD"]' is 'readonly' and cannot be assigned to the mutable type
'string[]'.
  Overload 2 of 2, '(units: Record<string, string |
UnitDefinition>, options?: CreateUnitOptions | undefined): Unit',
gave the following error.
    Argument of type 'string' is not assignable to parameter of
type 'Record<string, string | UnitDefinition>'.
      Type 'string' is not assignable to type 'Record<string,
string | UnitDefinition>'. ts(2769)
View Problem (Alt+F8)    No quick fixes available
            aliases: ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases});
        } else {
          // Sub unit (like: paise, cents etc.).
          createUnit(ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].name, {
            definition: ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].base_unit,
            aliases: ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases});
        }
```

The potential fix to the problem is to use Object.values() instead

```
  createCurrencyUnits(): void {
    var keys = <Array<keyof typeof ObjectsDomainConstants.CURRENCY_UNITS>>Object.keys(Object
    for (var i = 0; i < keys.length; i++) {
      if (ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].base_unit === null) {
        // Base unit (like: rupees, dollar etc.).
        createUnit(ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].name, {
          aliases: Object.values(ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases)});
      } else {
        // Sub unit (like: paise, cents etc.).
        createUnit(ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].name, {
          definition: ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].base_unit!,
          aliases: Object.values(ObjectsDomainConstants.CURRENCY_UNITS[keys[i]].aliases)});
      }
```

Situation 11:

A different situation which rises that involves “this” is as follows:

```
  'dom
                any
                'this' implicitly has type 'any' because it does not have
descri          a type annotation. ts(2683)
  desc          View Problem (Alt+F8)    Quick Fix... (Ctrl+.)
    be
      this.config = (
        new ExplorationImprovementsConfig('eid', 1, true
      this.createFromExplorationStats = (
          expStats: ExplorationStats, stateName: string,
          numEqPlaythroughs: number) => {
```

The way to handle this is by initializing the variables and their types and removing ‘this’ before each call.

```
27  describe('High bounce rate task model', function() {
28    let config: ExplorationImprovementsConfig;
29    let createFromExplorationStats: (
30      expStats: ExplorationStats,
31      stateName: string,
32      numEqPlaythroughs: number
33      )
34      => HighBounceRateTask;
35
36      beforeEach(() => {
37      config = (
38        new ExplorationImprovementsConfig('eid', 1, true, 0.25, 0.20, 100));
39      createFromExplorationStats = (
40          expStats: ExplorationStats, stateName: string,
```

Situation 12:

We should always change the spec file to adhere to the actual file and hence for the following null assertions, it is better to refactor the code

```
Input > directives > TS music-notes-input-validation.service.spec.ts > describe('MusicNotesInputValidationService') callback
47        feedback: {
48          html: '',
49          content_id: ''
50        },
51        labelled_as_correct: false,
52        param_changes: [],
53        refresher_exploration_id: null,
54        missing_prerequisite_skill_id: null
55      });
56      goodAnswerGroups = [agof.createNew([], goodDefaultOutcome, null, null
57    });
58

Argument of type 'null' is not assignable to parameter of type 'readonly InteractionAnswer[]'. ts(2345)
View Problem (Alt+F8)    No quick fixes available
```

Which can be done as follows:

```
55    });
56    goodAnswerGroups = [agof.createNew([], goodDefaultOutcome, [], '')];
57  );
58
```

Situation 13:

The below case demonstrates refactoring the code not in the test file

```
39  })
40  ex
41
42
43
44
45    currentAudioLanguageCode: string = null;
46    allAudioLanguageCodesInExploration: string[] = null;
47    explorationLanguageCode: string = null;
48    automaticTextToSpeechEnabled: boolean = null;
49    _languagesInExploration: ExplorationLanguageInfo[] = [];
50
51    attemptToSetAudioLanguageToExplorationLanguage(): void {
```

```
(property)
AudioTranslationLanguageService._currentAudioLanguageCode:
string

Type 'null' is not assignable to type 'string'. ts(2322)

View Problem (Alt+F8)    No quick fixes available
```

```
null-strict*   Python 3.6.9 64-bit   ⊗ 25 ⚠ 0   Spaces: 2   UTF-8   LF   TypeScript   4.2.3   Prettier
```

We can assign them the following initial values

```
45    _currentAudioLanguageCode: string = '';
46    _allAudioLanguageCodesInExploration: string[] = [];
47    _explorationLanguageCode: string = '';
48    _automaticTextToSpeechEnabled: boolean = false;
49    _languagesInExploration: ExplorationLanguageInfo[] = [];
50
```

Situation 14:

Consider the following situation where the return type may be undefined.

```
 * providing users
 * exploration.
 */
createNewWithState
  oldStateName:
 const newStateSt
 newStateStatsMap
  newStateName, this.stateStatsMapping.get(oldStateName));
 // ES2016 Map uses delete as a method name despite it being a reserved word.
 // eslint-disable-next-line dot-notation
```

```
(property) ExplorationStats.stateStatsMapping: ReadonlyMap<string,
StateStats>

Argument of type 'StateStats | undefined' is not assignable to
parameter of type 'StateStats'.
  Type 'undefined' is not assignable to type 'StateStats'. ts(2345)

View Problem (Alt+F8)    No quick fixes available
```

We can cater the condition of that state not existing and hence, ensure if undefined case arises, it will be handled.

```
125   createNewWithStateRenamed(
126       oldStateName: string, newStateName: string): ExplorationStats {
127     const newStateStatsMapping = new Map(this.stateStatsMapping);
128     const stateStats = this.stateStatsMapping.get(oldStateName)
129     if (!stateStats) {
130       throw new Error('no stats exist for old state: ' + oldStateName);
131     }
132     newStateStatsMapping.set(
133       newStateName, stateStats);
```

Third-party Libraries*

No third party Libraries need to be added.

Testing Approach

With the typescript strict mode enabled for a file, the compiler will throw errors if any strict rule is being violated. Also, with the *pre_push_hook.py* already present in the codebase, it is impossible for a developer (unix like systems) to push changes that fail the typescript checks. *typescript_checks.py* is the file for compiling and checking typescript. This is sufficient to ensure that the files that have strict mode enabled will pass the strict checks and since this mode will be enabled by default for any new file added to the codebase, the *pre_push_hook* ensures that they will be strictly typed. For systems where the hook does not work, the circleCI workflow jobs will fail if the checks do not pass.

Milestones

(The files have been grouped to 5 per PR and in a way to keep a balance between the level of complexity in each PR**)**

Milestone 1

Key Objective: All newly added files to the codebase are strictly typed and the set of 55 files and their tests (110 individual files) have typescript strict mode enabled and pass the typescript strict checks.

No.	**Description of PR**	**Prereq PR numbers**	**Target date for PR submission**	**Target date for PR to be merged**
1.1	Make all newly added files enforce TS strict checks	NA	26th May	1st June
1.2	Make TS checks strict for StoryContentsObjectFactory, RatingComputationService, SuggestionsService, TextInputTokenizer, WindowRef	NA	31st May	4th June
1.3	Make TS checks strict for RuleObjectFactory, UnitsObjectFactory, WrittenTranslationObjectFactory, TopicsAndSkillsDashboardPageService,	NA	16th June	19th June

	ThreadStatusDisplayService			
1.4	Make TS checks strict for StatesObjectFactory StateObjectFactory ComputeGraphService ShortSkillSummaryObjectFactory RubricObjectFactory	NA	18th June	21st June
1.5	Make TS checks strict for InteractionObjectFactory, ParamTypeObjectFactory, RecordedVoiceOverObjectFactory, NumberWithUnitsObjectFactory, ExtensionTagAssemblerService	NA	20th June	23rd June
1.6	Make TS checks strict for WinnowingPreprocessingService, PlaythroughIssueObjectFactory, PlayerCorrectnessFeedbackEnabledService, RatioExpressionInputRulesService + RatioModel, AlgebraicExpressionInputValidationService	NA	22nd June	25th June
1.7	Make TS checks strict for StoryObjectFactory UtilsService, SuggestionModalService, StoryReferenceObjectFactory, ReviewTestEngineService	1.2	24th June	27th June
1.8	Make TS checks strict for StateGraphLayoutService, VersionTreeService, ExplorationDiffService, VoiceoverObjectFactory, SolutionValidityService,	1.3, 1.4, 1.5	26th June	29th June
1.9	Make TS checks strict for DeviceInfoService, DocumentAttributeCustomizationService, DragAndDropSortInputValidationService, ExpressionSyntaxTreeService, FeedbackThreadObjectFactory + ThreadMessageModel	1.3	28th June	2nd July

1.10	Make TS checks strict for, AlertsService, BrowserCheckerService, CodeReplValidationService, ContinueValidationService, FractionInputRulesService	1.3, 1.7	1st July	4th July
1.11	Make TS checks strict for FocusManagerService, NumberWithUnitsValidationService, ParamSpecsObjectFactory, SidebarStatusService, SubtopicPageContentsObjectFactory	1.3, 1.5, 1.7	3rd July	7th July
1.12	Make TS checks strict for GraphUtilsService, LostChangeObjectFactory, MisconceptionObjectFactory, NumberAttemptsService, LearnerActionObjectFactory	1.3, 1.4, 1.5, 1.7, 1.8	4th July	7th July

Milestone 2

Key Objective: The set of 65 files and their tests (130 individual files) have typescript strict mode enabled and pass the typescript strict checks.

No.	Description of PR	Prereq PR numbers	Target date for PR submission	Target date for PR to be merged
2.0	Make the typescript types guide more thorough. (**wiki update**)	NA	9th July	NA
2.1	Make TS checks strict for FractionInputValidationService, GraphInputValidationService, HintObjectFactory, HtmlEscaperService, ImageClickInputValidationService	1.3	12th July	15th July
2.2	Make TS checks strict for GuestCollectionProgressService + CollectionNodeModel +	1.3, 1.5	14th July	17th July

	CollectionProgressModel, InteractiveMapValidationService, ItemSelectionInputValidationService, LocalStorageService			
2.3	Make TS checks strict for MathEquationInputValidationService, MessengerService, MetaTagCustomizationService, MultipleChoiceInputValidationService, MusicNotesInputRulesService	1.3, 1.7	16th July	19th July
2.4	Make TS checks strict for GraphInputRulesService, MusicNotesInputValidationService, NormalizeWhitespacePipe, NumericInputValidationService, NumericExpressionInputValidationService	1.3, 1.7, 1.12	20th July	23rd July
2.5	Make TS checks strict for NumberWithUnitsRulesService, OutcomeObjectFactory, PageTitleService, ParamChangesObjectFactory, ParamSpecObjectFactory	1.3, 1.5, 1.7	22nd July	25th July
2.6	Make TS checks strict for PencilCodeEditorValidationService, PlayerTranscriptService, AudioTranslationLanguageService, LanguageUtilService, StateCardObjectFactory	1.3	24th July	27th July
2.7	Make TS checks strict for PlaythroughObjectFactory, PythonProgramTokenizer, QuestionsObjectFactory, RatioExpressionInputValidationService, SetInputValidationService	1.3, 1.6	26th July	29th July
2.8	Make TS checks strict for SkillCreationBackendApiService, StateTopAnswersStatsObjectFactory, SpeechSynthesisChunkerService, SVMPredictionService, SchemaDefaultValueService	NA	28th July	31st July
2.9	Make TS checks strict for	1.3,	30th July	2nd August

	SiteAnalyticsService, StateClassifierMappingService, StoryContentsObjectFactory, SubtopicObjectFactory, TextInputRulesService	1.7,		
2.10	Make TS checks strict for TopicCreationBackendApiService, UrlService, WindowDimensionsService, WorkedExampleObjectFactory, AnswerGroupObjectFactory	1.3	1st August	4th August
2.11	Make TS checks strict for CkEditorCopyContentService, AutogeneratedAudioPlayerService, BottomNavbarStatusService, PreventPageUnloadEventService, ValidatorsService	1.7, 2.8	3rd August	6th August
2.12	Make TS checks strict for PlayerTranscriptService, QuestionValidationService, StateEditorService, TextInputValidationService, ExpressionSyntaxTreeService	1.3, 1.4, 1.5, 1.8, 2.6, 2.7, 2.9	5th August	8th August
2.13	Make TS checks strict for CodeReplPredictionService, CodeReplRulesService, ConceptCardObjectFactory, EditorFirstTimeEventsService, TopicObjectFactory	1.5, 1.6, 1.7, 2.8, 2.9	8th August	11th August

Optional Sections

Additional Project-Specific Considerations

Please elaborate on the specific documentation changes that will be added in this project. Be as concrete as possible, and provide clear details.

Documentation Changes:

The following documentation changes may be required:

- An assertion to developers that TS strict typing will be enabled for every new file to be added to the codebase.
- A guide on typescript types is already present in Oppia wiki. This needs to be made more thorough with additional description (with examples) of the rules that are enforced due to strict mode.

The Guide on Defining Types will be updated with the addition of the following section and subsections:

TypeScript Strict Mode:

(This is just a draft of the expected changes to the guide, the updated guide will be more detailed)

Why Enable Strict Mode?

This section will contain an elaboration on the following points:

- Provides self documentation
- Catches edge cases and reduces potential runtime errors
- Enables writing more robust code

The Strict Rules:

*(This will be a general explanation of the rules with simple code examples **alongside explanations of the code snippets**, not targeted towards Oppia codebase)*

noImplicitAny

Definition:

This rule disallows variables and function arguments to have implicit type `any`

Violation:

```
const multiply2 = (num) => num * 2;
// error: parameter 'num' implicitly has an 'any' type
```

Potential Fix:

```
const multiply2 = (num: number) => num * 2;
```

noImplicitThis

Definition:

This rule disallows the context of `this` to be defined implicitly

Violation:

```
class Employee {
  constructor(private name: string) {}

  greetEmployee() {
    return function() {
      console.log(`Hello ${this.name}!`);
      // error: 'this' implicitly has type 'any' because it does not have a type annotation.
    };
  }
}
```

Potential Fix:

```
greetEmployee() {
  return function (this: Employee) {
    ...
  };
}
```

or

```
greetEmployee() {
  return () => {
    ...
  };
}
```

strictNullChecks

Definition

Values can be null or undefined only if explicitly marked

Violation - undefined:

```
function getEmployeeById (employees: Employee[], id: string): Employee {
  const employee = employees.find(employee => employee.id === id);
  return employee;
  // error:  Object is possibly 'undefined'.
}
```

Potential Fix:

```
function getEmployeeById (employees: Employee[], id: string): Employee {
```

```
  const employee = employees.find(employee => employee.id === id)
  if (typeof employee === 'undefined') {
    throw new Error(`Could not find employee with id: ${id}.`);
  }
  return employee;
}
```

Violation - null:

```
let employee: Employee;
...
employee = null;
// error: type null is not assignable to type ‘Employee’
```

Potential Fix:

```
let employee: Employee;
...
employee = new Employee();

Or

let employee: Employee | null;
...
employee = null;
```

strictPropertyInitialization

Definition

All class properties need to be initialized in a constructor or property initializer

Violation:

```
class Employee {
  name: string;
  constructor(private name: string) {}

// error: Property 'name'' has no initializer and is not definitely assigned in the constructor
```

Potential Fix:

```
class Employee {
  name: string;
  constructor(private name: string) {
      this.name = name;
```

```
}
```

strictBindCallApply

Definition

Enforces stricter checking of `bind`, `call` and `apply` functions

Violation:

```
calculatePay: (bonusCode: string, price: number) => number;
...
const totalPay = employee.calculatePay.apply(
  undefined,
  ["CIT"]
);

/*
The call to calculatePay above is violating this rule because calculatePay requires two
arguments.
*/
```

Potential Fix:

```
const totalPay = employee.calculatePay.apply(
  undefined,
  ["CIT", 15000]
);
```

strictFunctionTypes

Definition:

Argument types cannot be bivariant

Violation:

```
const getEmployeeByName = (name: string) => {
    employee.find((employee) => employee.name == name)
}

type getEmployeeByNameType = (name: string | number) => void;

const fn: getEmployeeByNameType = getEmployeeByName;

// error: Type '(name: string) => void' is not assignable to type 'getEmployeeByNameType'.
```

Potential Fix:

```
type getEmployeeByNameType = (name: string | number) => void;

const getEmployeeByName: getEmployeeByNameType = (name: string) => {
    employee.find((employee) => employee.name == name)
}

const fn: getEmployeeByNameType = getEmployeeByName;
```

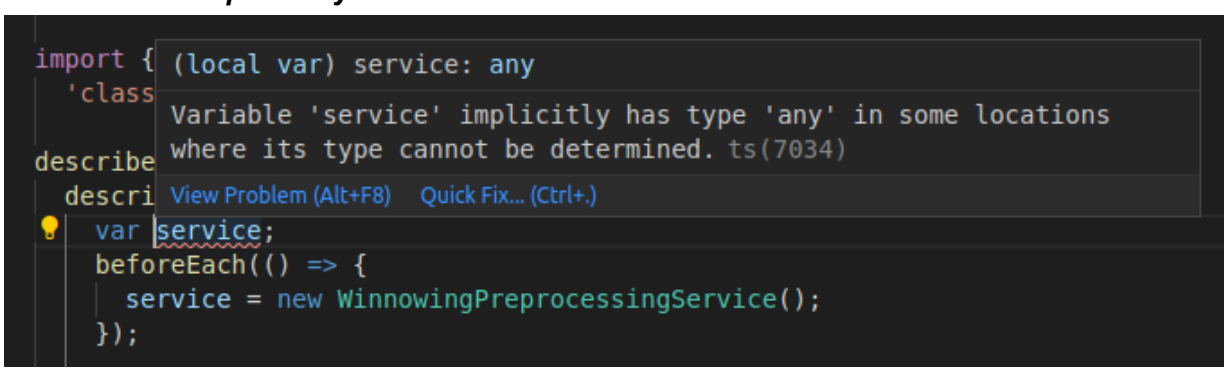
Cases Encountered in Oppia Codebase:

*(This will include the list of specific recurring situations which came into view throughout the GSoC tenure and their **approved** fixes.)*

Case 1:

Violation:

The type of variable is not defined explicitly and hence typescript assigns it an '*any*' type. This violates the ***noImplicitAny*** rule of strict mode.



Solution:

To solve this, the variable is explicitly assigned the type it belongs to. In some cases in the Oppia codebase, we may need to import the Interface, Class or Type as it is not imported in the file before.
In this example, the service was of type *WinnowingPreprocessingService*.

```
describe('Winnowing preprocessing functions', () => {
  describe('Test winnowing preprocessing functions', () => {
    var service: WinnowingPreprocessingService;
    beforeEach(() => {
      service = new WinnowingPreprocessingService();
    });
```

Privacy

This project involves adding checks to improve the code by either adding additional types to the code or refactoring the code to avoid adding a new type altogether. It, hence, does not change the functionality of the code and nor does it change the way user data is collected. There are no privacy considerations to be taken into account.

Security

This project has no security concerns nor brings in any opportunity for the developers to gain unauthorized access to any part of the code or user data. This is due to the fact that the changes involve refactoring the code and there is no addition of any new functionality or feature that may breach security.

Accessibility (if user-facing)

The users of this project are the developers of Oppia and when it comes to accessibility, adding types itself makes the code base self documenting and easier to understand and comprehend. Also, additional documentation will be added about strict typing and will serve as a guide for developers who may not be introduced to strict typing and its rules before.

Ethics*

Since all this project involves is making the code more robust by introducing strict typing to the code. There is no additional feature being introduced that might need any ethical considerations to be taken into account.

Future Work

The project involves enabling strict mode for a limited number of files and hence, does not introduce strict typing to the entire code base of Oppia. After GSoC, the remaining files could be introduced to strict typescript checks so that two tsconfig files may not be needed.