Code Documentation
Team 8

API Controller

Scripts:

- **apiController.gd (main)**

[SCRIPT] apiController.gd

Description

apiController.gd serves as the HTTP REQUEST controller to communicate with the Web API and to retrieve relevant data from it.

Properties

Data Type	**Property Name**
String	baseUrl
Array<String>	result
int	response_code

Property Descriptions

- **String** baseUrl
  Base URL Route to the WEB API. Defaulted to **“https://learnez.a2hosted.com/public/api/”**

- **Array<String>** result
  JSON data retrieved from the WEB API.

- **int** response_code
  Status of the HTTP Request Call. Returns “**200”** if successful **and “404”** if unsuccessful.

Methods

Return Type	Method Name
void	apiCallGet (String url)
void	apiCallPut (Array<String> data , String url)
void	apiCallPost Array<String> data, String url)

void	on_HTTPRequest_request_completed (result, response_code, headers, body)

Methods Description

- **void** apiCallGet (String url)
  Make an HTTP Get Request call to the WEB API to retrieve data from the server's database. *url* refers to the WEB API Route to make the HTTP GET Request Call.

- **void** apiCallPut (Array<String> data, String url)
  Make an HTTP Put Request call to the WEB API to update data in the server's database. *data* refers to the JSON payload to be posted and the url refers to the WEB API Route to make the HTTP PUT Request Call.

- **void** apiCallPost (Array<String> data, String url)
  Make an HTTP Post Request call to the WEB API to create data in the server's database. *data* refers to the JSON payload to be posted and the url refers to the WEB API Route to make the HTTP POST Request Call.

- **void** on_HTTPRequest_request_completed (result, response_code, headers, body)
  Invoked after an HTTP GET/PUT/POST Request has been made to the WEB API and the request has been successfully completed. body contains the JSON data that is being retrieved from the WEB API. response_code contains the status of the HTTP Request Call.

Global Controller

Scripts

- **global.gd** (main)

[SCRIPT] global.gd

Description

global.gd is used for handling the navigation flow between pages as well as the screen's orientation (portrait/landscape) of the different pages.

Properties

Data Type	**Property Name**
String	current_scene_path
String	root_scene
Array<String>	history

Property Descriptions

- **String** current_scene_path
  Current URL of the Page that the User is on.

- **String** root_scene
  URL of the Landing Page.

- **Array<String**> history
  The navigation history (sequence of visited Pages) of the User.

Methods

Return Type	Method Name
void	switch_scene (String path)
void	return_to_last()
void	set_sceen_orientation (value)

Methods Description

- **void** switch_scene (String path)

Changes the Page of the application based on the url provided by the `path`.

- **void** return_to_last()
  Returns to the previous page based on the User's Navigation History.

- **void** set_screen_orientation (value)
  Sets the default screen orientation for the Page. "Portrait Mode" if `value` is 1, "Landscape Mode" if `value` is 0

Login Controller

Scripts:

- loginController.gd
- **login_button.gd (Main)**
- **reset_btn.gd (Main)**
- loginController.gd
    - Description: Handles Show/Hide of UI Elements
- forget_password.gd
    - Description: Handles Forget Password Popup Box
- forget_pass_sucess_popup.gd
    - Description: Handles Popup Box for Successful Forget Password
- noAccount_popup.gd
    - Description: Handles Popup Box for Login Error

[SCRIPT] login_Btn.gd

Description

login.gd is used for handling users' login into the LearnEz's application.

Properties

Data Type	**Property Name**
String	url
Array<String>	result
int	responseCode
Object	usernameLabel
Object	passwordLabel
Object	loading_bg
Object	loading_sprite

Property Descriptions

- **String** url
  URL Extension Route for WEB API

- **Array<String>** result

JSON data retrieved from the WEB API.

- **int** responseCode
  Status of the HTTP Request Call.

- **Object** usernameLabel
  "Username Field is empty"" Label.

- **Object** username Label
  "Password Field is empty"" Label.

- **Object** loading_bg
  Background for Loading Screen

- **Object** loading_sprite
  Image for Loading Screen

Methods

Return Type	Method Name
void	verifyUser()
void	getUserInfo()

Methods Description

- **void** verifyUser()
  Verifies User's Input fields (User ID and Password). If fields are not empty, make API call to get User's Data. Triggers `noAccount_Popup.gd` if User's Data cannot be found. Invokes getUserInfo() otherwise.

- **Void** getUserInfo()
  Updates User's Model with retrieved User's Data. Check for User's Role and redirects to the respective Main Menu Page.

[SCRIPT] reset_Btn.gd

Description

reset.gd is used for resetting User's Password

Properties

Data Type	Property Name
String	url
Array<String>	result
int	responseCode
Object	resetByIDLbl
Object	resetPasswordErr

Property Descriptions

- **String** url
  URL Extension Route for WEB API

- **Array<String>** result
  JSON data retrieved from the WEB API.

- **int** responseCode
  Status of the HTTP Request Call.

- **Object** resetByIDLbl
  User's Input Field (Matriculation Number / Staff ID).

- **Object** resetPasswordErr
  "User ID Field is empty"" Label.

Methods

Return Type	Method Name
void	resetPassword()

Methods Description

- **void** resetPassword()
  Verifies User's Input field (User ID). If field is not empty, make API call to get User's Data. Shows resetPasswordErr if User's Data cannot be found. Send Reset Password link to user's registered email otherwise.

Settings Controller

Scripts:

- **change_password_box.gd (Main)**
- logout_box.gd
    - Renders the logout scene, as well as button logic
- password_updated_box.gd
    - Renders the pop up message to inform user of successful/unsuccessful password change, as well as register when user closes the pop up message.
- SettingsController.gd
    - Handles the processing and displaying of relevant user's information on the settings page.

[SCRIPT] change_password_box.gd

Description

Change_password_box.gd handles the password changing process, including validation of user input, and database update.

Properties

Data Type	Property Name
String	url
Array<String>	result
int	responseCode

Property Descriptions

- **String** url
  URL Extension Route for WEB API

- **Array<String>** result
  JSON data retrieved from the WEB API.

- **int** responseCode
  Status of the HTTP Request Call.

Methods

Return Type	Method Name
void	verifyPasswordInput

Methods Description

- **verifyPasswordInput()**
  Verifies User's Input fields (Password & Re-Enter Passwords). If fields are empty or do not match, show error message. Otherwise, make API call to update User’s Password based on User’s Input.

Avatar Controller

Scripts:

- **confirmation_popup.gd (Main)**
- AvatarController.gd
    - Description: Handles user selection of Avatar and display of Selected Avatar

[SCRIPT] confirmation_popup.gd

Description

confirmation_popup.gd is used for updating User's Avatar.

Properties

Data Type	**Property Name**
String	apiUrl
Array<String>	result
int	responseCode

Property Descriptions

- **String** url
  URL Extension Route for WEB API

- **Array<String>** result
  JSON data retrieved from the WEB API.

- **int** responseCode
  Status of the HTTP Request Call.

Methods

Return Type	Method Name
void	closeConfirmationDialog()
void	handleChangeAvatar()

Methods Description

- **closeConfirmationDialog()**
  Hides the Avatar Confirmation Popup.

- **handleChangeAvatar**
  Make API Call to update User’s Avatar based on selected Avatar.

User Model

Scripts:

- **userModel.gd (Main)**

[SCRIPT] userModel.gd

Description

userModel.gd is used for storing information of a User

Properties

Data Type	Property Name
String	baseUrl
String	userID
String	userEmail
String	userName
String	userPassword
String	userRole
String	userAvatarID
String	userAvatar
String	userGroup
int	userCurrency

Property Descriptions

- **String baseUrl**

Setter	setBaseUrl (url)
Getter	getBaseUrl()

- **String userID**

Setter	setUserID (userID)

Getter	getUserID()

- **String userEmail**

Setter	setUserEmail (userEmail)
Getter	getUserEmail()

- **String userName**

Setter	setUserName (userName)
Getter	getUserName()

- **String userPassword**

Setter	setUserPassword (userPassword)
Getter	getUserPassword ()

- **String userRole**

Setter	setUserRole (userRole)
Getter	getUserRole)

- **String userAvatarID**

Setter	setUserAvatarID (userAvatarID)
Getter	getUserAvatarID()

- **String userAvatar**

Setter	setUserAvatar (userAvatar)
Getter	getUserAvatar()

- **String userGroup**

Setter	setUserGroup (userGroup)
Getter	getUserGroup ()

- **String userCurrency**

Setter	setUserCurrency (userCurrency)

Getter	getUserCurrency(userCurrency)

View UCL Controller

Scripts:

- **UCLView.gd (Main)**

[SCRIPT] UCLView.gd

Description

UCLView.gd handles the fetching and displaying of user created levels, as well as the user's selected user created level. It also handles the logical navigation of the page and can redirect the user to create a new level.

Properties

var apiUrl_UCL = "ucl/all/all"
var result_UCL
var responseCode_UCL
var base_index = 0
var max_index
var nextPage = "true"
var number_to_minus = 0
var pageNumber = 1
var returnResult = []

Data Type	**Property Name**
String	apiUrl_UCL
Array<String>	result_UCL
String	responseCode_UCL
int	base_index
int	max_index
Boolean	nextPage
int	number_to_minus
int	pageNumber
int	returnResult

Property Descriptions

- **String** apiUrl_UCL
  URL used for fetching all UCLs from database.
- **Array<String>** result_UCL
  This is an array of dictionary, fetched from database. Each dictionary in the array is a question in a user created level.
- **String** responseCode_UCL
  Response code used for the developer to troubleshoot.
- **int** base_index
  Used for checking the current page, and for ensuring smooth navigation. It starts at 0.
- **int** max_index
  Used for ensuring the user cannot navigate to a page where there are no UCLs to be displayed.
- **Boolean** nextPage
  A boolean signal to determine whether the next page option should be displayed.
- **int** number_to_minus
  This is used together with base_index for navigational purposes.
- **int** pageNumber
  pageNumber is an integer used to ensure that the user's selected UCL can be found in result_UCL.
- **int** returnResult
  An integer for the user's chosen UCL index within the result_UCL array.

Methods

Return Type	Method Name
void	render_ucl

Methods Description

- **render_ucl**
  Function that is called whenever the next page or previous page button is pressed. Also a function called when the scene loads. It handles displaying the correct UCL, in the correct order.

Create UCL Controller

Scripts:

- **CreateUCLController.gd**

[SCRIPT] CreateUCLController.gd

Description

This controller handles the loading, as well as creation of UCL logic.

Properties

Data Type	**Property Name**
String	url
Array<String>	result_UCL
int	responseCode_UCL

Property Descriptions

- **String** url
  The url used to fetch all UCL from database.
- **Array<String>** result_UCL
  An array of UCL information.
- **int** responseCode_UCL
  Used by the developer to check the response code of the database request.

Methods

Return Type	Method Name
void	create_Pressed

Methods Description

- **create_Pressed()**
  Handles the logic for creating a user created level, in the database.

UCL Model

Scripts:

- **uclModel.gd (Main)**

[SCRIPT] uclModel.gd

Description

uclModel.gd is used for storing information of all User Created Levels.

Properties

Data Type	Property Name
String	baseUrl
int	selectedUCL
String	selectUCLIdx
String	ucl

Property Descriptions

- **String baseUrl**

Setter	setBaseUrl (url)
Getter	getBaseUrl()

- **String selectedUCL**

Setter	setSelectedUCL(UCLId)
Getter	getSelectedUCL(UCLId)

- **Array<String> selectUCLIdx**

Setter	setSelectedUCLIdx(idx)

- **Array<String> ucl**

Setter	setUCL (ucl)

Getter	getNumberOfUCL()
Getter	getUCLByIdx(idx)
Getter	getUCLNameByIdx(idx)

Game Controller

Scripts:

- **Game.gd (Main)**
- **Map.gd (Main)**
- **Player.gd (Main)**
- **gameClearPopup.gd (Main)**
- **Questions.gd (Main)**
- **playermodel.gd**
- **Timer.gd**
- **TimeUpPopUp.gd (Main)**

[SCRIPT] Map.gd

Description

Game.gd is used to control the game's map and enemy generation.

Methods

Return Type	Method Name
void	_ready
void	loadQuestions

Methods Description

- **_ready**
  Function is called during scene transition into Game. _ready handles generation of 'Time Up' button generation and population of gameModel environmental variables.
- **loadQuestions**
  Function is called during _ready. loadQuestions generates the question list through connection with the APIController to load questions from the server database. It populates gameModel with a list of questions with difficulty rating 1-10.

[SCRIPT] Map.gd

Description

Map.gd is used to control the game's map and enemy generation.

Properties

Data Type	**Property Name**

int	tile_size
int	width
int	height
Vector2 []	enemyloc
cell_walls	dict
const int	N
const int	E
const int	S
const int	W

Property Descriptions

- **int tile_size**
  Determines the length of the square tile edges in pixels.
- **int width**
  Determines the number of tiles in the width of the TileMap used to generated the maze.
- **int height**
  Determines the number of tiles in the height of the TileMap used to generated the maze
- **Vector2[] enemyloc**
  Array of vectors used to determine existing enemy locations

- **dict cell_walls**
  Dictionary used to determine representation of directions in each tile. The keys are the unit vectors of direction mapped to the corresponding string.
- **const int N**
  Bit representing presence of wall in north direction.
- **const int E**
  Bit representing presence of wall in east direction.
- **const int S**
  Bit representing presence of wall in south direction.
- **const int W**
  Bit representing presence of wall in west direction.

Methods

Return Type	Method Name
void	_ready

void	_physics_process
list	check_neighbours
void	make_maze
list	shuffleList
void	make_enemy
void	generate_enemies

Methods Description

- **_ready**
  Function is called during scene transition into Game. _ready handles generation of map using make_maze and generation of enemies using generate_enemies.
- **_physics_process**
  Function is called during every physics calculation step of the Game. _physics_process handles the detection of collision between player object and enemy objects, and subsequent triggers of quizzes or completion. Collision between player and terrain is handled through the game engine and not explicitly defined in this function.
- **check_neighbours**
  Function is called during make_maze to check the unvisited neighbours of a particular cell. It returns a list of unvisited cells.
- **make_maze**
  Function is called to procedurally generate the maze environment using recursive backtracking. It directly edits the TileMap object within the game environment. It uses int width and int height to determine the size of maze that will be generated.
- **shuffleList**
  Function is called to shuffle a list. As Godot does not have an inbuilt list shuffling function, shuffleList is implemented.
- **make_enemy**
  Function is called to place enemy object on the map and create corresponding enemy object.
- **generate_enemy**
  Function is called to generate enemies after the procedurally generated maze is created. The enemies position is randomly selected within the size of the maze using int width and int height. The enemies position will never be at the start of the maze. Generate_enemy uses make_enemy to place the enemy object into the engine's environment

[SCRIPT] Player.gd

Description

Player.gd is used to control the player movement within the game area.

Properties

Data Type	**Property Name**
int	SPEED
Vector2	velocity
Vector2	screen_size

Property Descriptions

- **int SPEED**
  Determines the maximum speed of the player.
- **Vector2 velocity**
  Determines the current velocity of the player.
- **Vector2 screen_size**
  Determines the maximum play area of the player.

Methods

Return Type	Method Name
void	_ready
void	_physics_process

Methods Description

- **_ready**
  Function is called during scene transition into Game. _ready handles the collection of the current device's screen size.
- **_physics_process**
  Function is called during every physics calculation step of the Game. _physics_process handles player movement and clamps the position of the player to never exceed the screen size. Player movement is 8-way.

[SCRIPT] gameClearPopup.gd

Description

gameClearPopup.gd is used to control the game's completion

Methods

Return Type	Method Name
void	updateUserGameClear():

Methods Description

- **updateUserGameClear**
  updateUserGameClear is used to update the server database when the player completes the game successfully. updateUserGameClear makes a database entry with the stage of the game and the relevant score.

[SCRIPT] Questions.gd

Description

Questions.gd is used to control question fetching.

Properties

Data Type	**Property Name**
int	randomQuestionIdx
JSON array	currQuestion
int	correctOption

Property Descriptions

- **int randomQuestionIdx**
  Determines the index of which to fetch the question.
- **JSON array currQuestion**
  Stores the question description, options and correct answers for each question.
- **Int correctOption**
  Determines the index of the correct option.

Methods

Return Type	Method Name
void	_ready
void	loadCurrentQuestion
void	useFiftyfifty()

Methods Description

- **_ready**
  Function is called during scene transition into question. _ready obtains the number of ‘50-50’ powerups and enables their use.
- **loadCurrentQuestion**
  Function is called during _ready to load the question to be used in the question popup. loadCurrentQuestion access the gameModel object to randomly generate a question of the relevant difficulty using int randomQuestionIdx and external method getQuestionsByDifficulty.
- **useFiftyFifty**
  Function is called whenever ‘50-50’ is used. Function handles controller logic to ensure correct use of the powerup.

[SCRIPT] TimeUpPopUp.gd

Description

TimeUpPopUp.gd is used to prompt the user when their time is up in the maze.

Properties

Data Type	**Property Name**
int	ms
int	s
int	m
boolean	running
signal	no_time

Property Descriptions

- **int ms**
  Determines the time of the game.
- **int s**
  Stores the second of the timer.
- **int m**
  Stores the minute of the timer.
- **boolean running**
  Check if the timer is still running.
- **signal no_time**
  Check if the timer has run out of time.

Methods

Return Type	Method Name
void	_process
void	_on_ms_timeout

Methods Description

- **_process**
  Function is called during the start of the game. _process will obtain the total time for the timer for the game that is being played. It also checks if the timer has run out of time and return a signal of timeout.
- **_on_ms_timeout**
  Function is called to decrease the timer as the timer is counting down

Level Model

Scripts:

- **levelModel.gd**

[SCRIPT] levelModel.gd

Description

Acts as the model for storing all level information.

Properties

Data Type	**Property Name**
String	baseUrl
int	selectedLevel
int	selectedLevelIdx
Array<Object>	levels
int	unlockStatus

Property Descriptions

- **String baseUrl**

Setter	setBaseUrl (url)
Getter	getBaseUrl()

- **String selectedLevel**

Setter	setSelectedLevel (levelID)

- **String selectedLevelIdx**

Setter	setSelectedLevelIdx(idx)

- **Array<Object> levels**

Setter	setLevels(levels)

Getter	getLevelByIdx(idx)
Getter	getNumberOfLevels()
Getter	getLevelNameByIdx(idx)
Getter	getLevelStageByIdx(idx)
Getter	getLevelDescription(idx)

- **int unlockStatus**

Setter	setUnlockStatus(status)
Getter	getLevelScoreByIdx(idx)
Getter	getLevelUnlockStatusByIdx(idx)

`

Powerup Model

Scripts:

- **powerupModel.gd (Main)**

[SCRIPT] powerupModel.gd

Description

powerupModel.gd is used for storing information of powerup information

Properties

Data Type	**Property Name**
String	baseUrl
Array<Object>	powerups
int	selectedPowerupIdx
int	selectedPowerup

Property Descriptions

- **String baseUrl**

Setter	setBaseUrl (url)
Getter	getBaseUrl()

- **Array<Object> powerups**

Setter	setPowerups(powerups)

- **Object selectedPowerupIdx**

Setter	setSelectedPowerupIdx(idx)
Getter	getSelectedPowerupIdx()

- **Object selectedPowerup**

Setter	setSelectedPowerup()

Getter	getSelectedPowerupCost()
Getter	getSelectedPowerupName()
Getter	getSelectedPowerupID()

Assignment Board Controller

Scripts:

- **AssignmentController.gd (Main)**
- **No_Assignment_Popup.gd**
- **Post_Assignment_Sucess_Popup.gd**
- **PostAssignment.gd (Main)**

[SCRIPT] AssignmentController.gd

Description

This controller handles the viewing of the assignment.

Properties

Data Type	**Property Name**
String	no_assignment_background
String	default_background
String	apiUrl
String	apiUrl_student
String	apiUrl_teacher_group
String	apiUrl_teacher_all
Array<String>	result
int	responseCode
String	selectedClass
boolean	select_class
int	base_index
int	max_index
boolean	go_left
boolean	go_right

Property Descriptions

- **String no_assignment_background**

  An image that is displayed when there is no assignment available

- **String default_background**

  An image background that is displayed when there is an assignment available

- **String apiUrl**

  URL Extension Route for WEB API

- **String apiUrl_student**

  URL Extension Route for WEB API for student

- **String apiUrl_teacher_group**

  URL Extension Route for WEB API for teacher groups

- **String apiUrl_teacher_all**

  URL Extension Route for WEB API for teacher

- **String result**

  An array of Assignment Controller results

- **Int response**

  Status of the HTTP Request Call

- **String selectedClass**

  Selected class in the teacher’s groups

- **boolean select_class**

  Fetch the specified class or retrieve all the class group for assignment

- **int base_index**

  Using a base number to move back and forth between the assignment pages

- **int max_index**

  Maximum allowable pages for the assignment page to go until

- **boolean go_left**

  Check whether if there is any previous page of assignments from current page

- **boolean go_right**

  Check whether if there is any next page of assignments from current page

Methods

Return Type	Method Name
void	render_assignment()

Methods Description

- **render_assignment()**
  Handles the rendering of the assignment page.

[SCRIPT] PostAssignment.gd

Description

This controller handles the posting of the assignment.

Properties

Data Type	**Property Name**
String	url
String	apiUrl_file
String	result
String	result_fileURL
int	responseCode
String	responseCode_URL
String	selectedClass
String	selectedFile
boolean	title_valid
boolean	details_valid
boolean	class_valid

boolean	file_valid
boolean	year_valid
boolean	month_valid
boolean	day_valid
boolean	minute_valid
boolean	hour_valid

Property Descriptions

- **String url**

  URL Extension Route for WEB API for teacher

- **String apiUrl_file**

  URL Extension Route for WEB API for assignment file

- **String result**

  An array of Assignment Controller results

- **String result_fileURL**

  URL Extension Route for WEB API for resulting assignment file URL

- **int responseCode**

  Used by the developer to check the response code of the database request

- **String selectedClass**

  Selected class in the teacher’s groups

- **String selectedFile**

  Selected file in the teacher’s groups

- **boolean title_valid**

  Check if assignment title is valid

- **boolean details_valid**

  Check if assignment detail is valid

- **boolean class_valid**

Check if class is valid

- **boolean file_valid**

  Check if file is valid

- **boolean year_valid**

  Check if year is valid

- **boolean month_valid**

  Check if month is valid

- **boolean day_valid**

  Check if day is valid

- **boolean hour_valid**

  Check if hour is valid

- **boolean minute_valid**

  Check if minute is valid

Methods

Return Type	Method Name
void	post_assignment_pressed()

Methods Description

- **post_assignment_pressed()**
  Handles the posting of the new assignment.

World Model

Scripts:

- **worldModel.gd (Main)**

[SCRIPT] worldModel.gd

Description

worldModel.gd is used for storing information of a World

Properties

Data Type	**Property Name**
String	baseUrl
int	selectedWorld
int	selectWorldIdx
Array<Object>	worlds
int	unlockStatus

Property Descriptions

- **String baseUrl**

Setter	setBaseUrl (url)
Getter	getBaseUrl()

- **int selectedWorld**

Setter	setSelectedWorld(worldID)

- **int selectWorldIdx**

Setter	setSelectedWorldIdx(idx)

- **Array<Object> worlds**

Setter	setWorlds(worlds)

Getter	getNumberOfWorlds()
Getter	getWorldByIdx(idx)
Getter	getWorldNameByIdx(idx)
Getter	getWorldIDbyIdx(idx)

- **int unlockStatus**

Setter	setUnlockStatus(status)

Discussion Board Controller

Scripts:

- DetailedDiscussion.gd
- **DiscussionBoardController.gd (Main)**
- **PostDiscussion.gd (Main)**
- Post_Discussion_Sucess_Popup.gd

[SCRIPT] DiscussionBoardController.gd

Description

This controller handles the logic for the discussion board view.

Properties

Data Type	**Property Name**
String	apiUrl_discussion
Array<Object>	result_discussion
int	responseCode_discussion
int	base_index
int	max_index
Boolean	go_right
int	number_to_minus

Property Descriptions

- **String** apiUrl_discussion
  Used for fetching discussions from database.
- **Array<Object>** result_discussion
  An array of discussion objects stored after fetching from database
- **int** responseCode_discussion
  A response code for the developer to do checks.
- **int** base_index
  Used for navigational purposes for multi pages.
- **int** max_index
  Used to determine how many pages there should be.
- **Boolean** go_right
  Navigational logic, if true button appears, else it will hide.

- **int** number_to_minus
  Used for navigational logic.

Methods

Return Type	Method Name
void	discussion_clicked (meta)

Methods Description

- **discussion_clicked (meta)**
  Handles the logic when the user selects a discussion. This method is repeated for all buttons in discussion page.

[SCRIPT] PostDiscussion.gd

Description

This controller handles posting discussion logic.

Properties

Data Type	**Property Name**
String	url

Property Descriptions

- **String** url
  Used for fetching discussions from database.

Methods

Return Type	Method Name
void	post_discussion_pressed()

Methods Description

- **post_discussion_pressed (meta)**
  Handles the logic when a discussion is attempted to be posted.