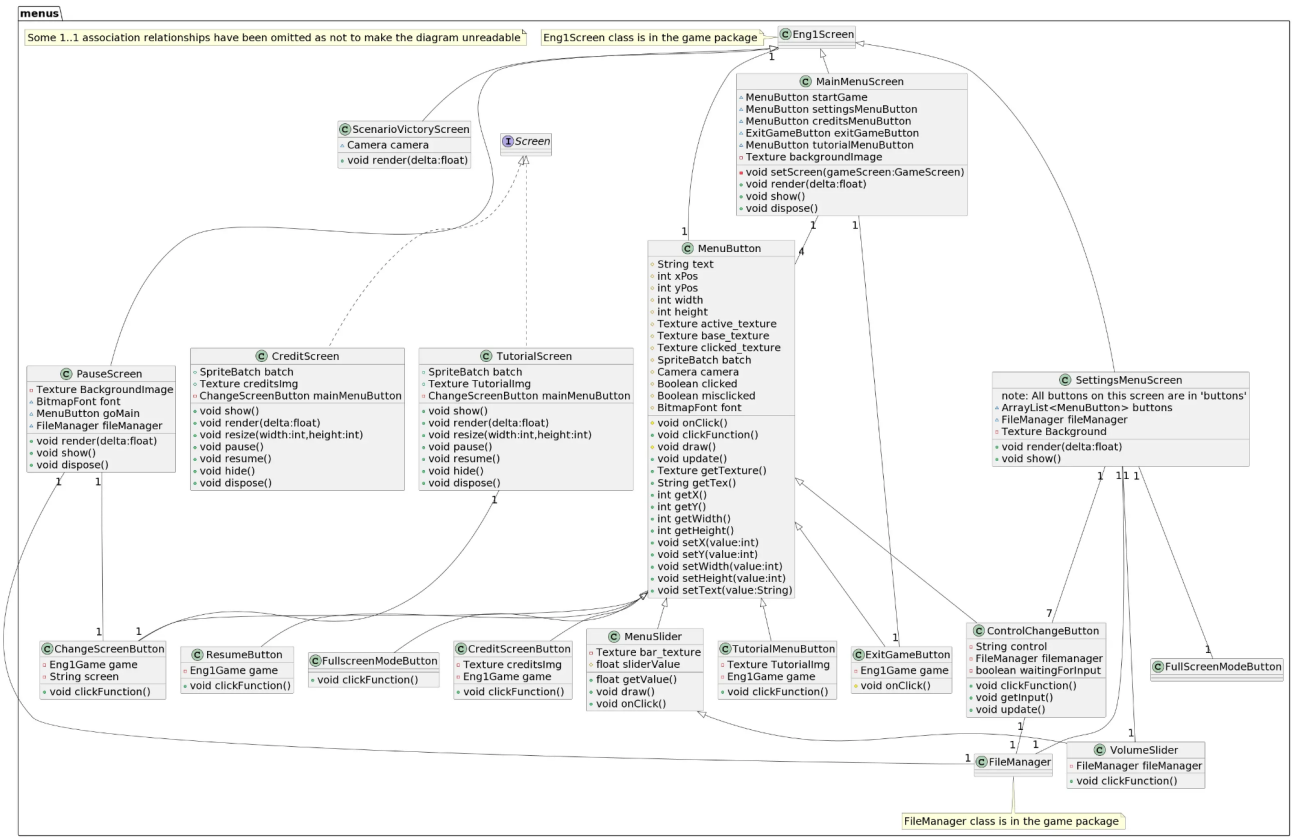
Architecture

Group Number: 10

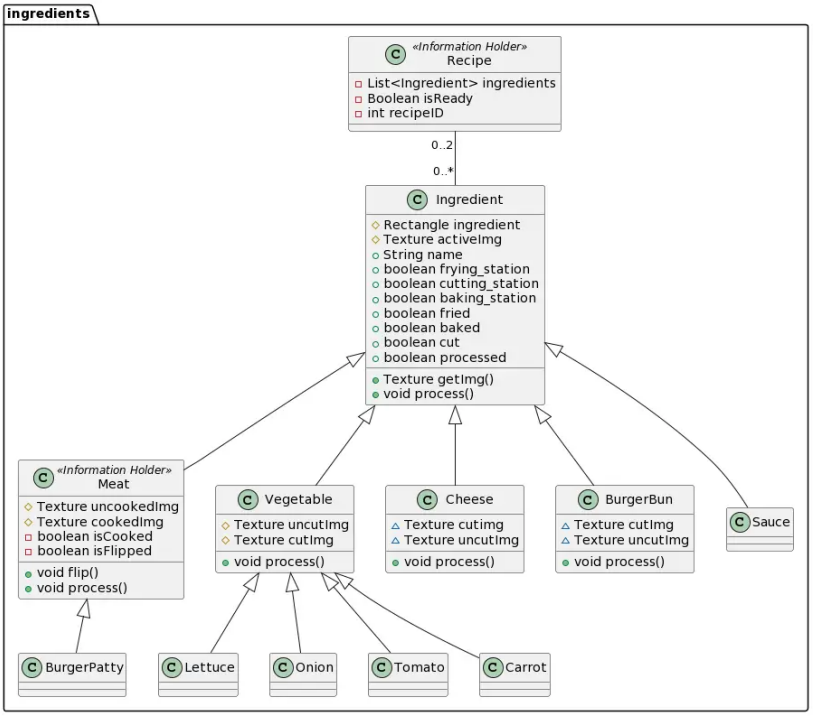
Team Name: Decassociation

Group Member Names:
Mohammad Abdullah
Tom Broadbent
Poppy Fynes
Owen Lister
Michael Marples
Lucy Walsh

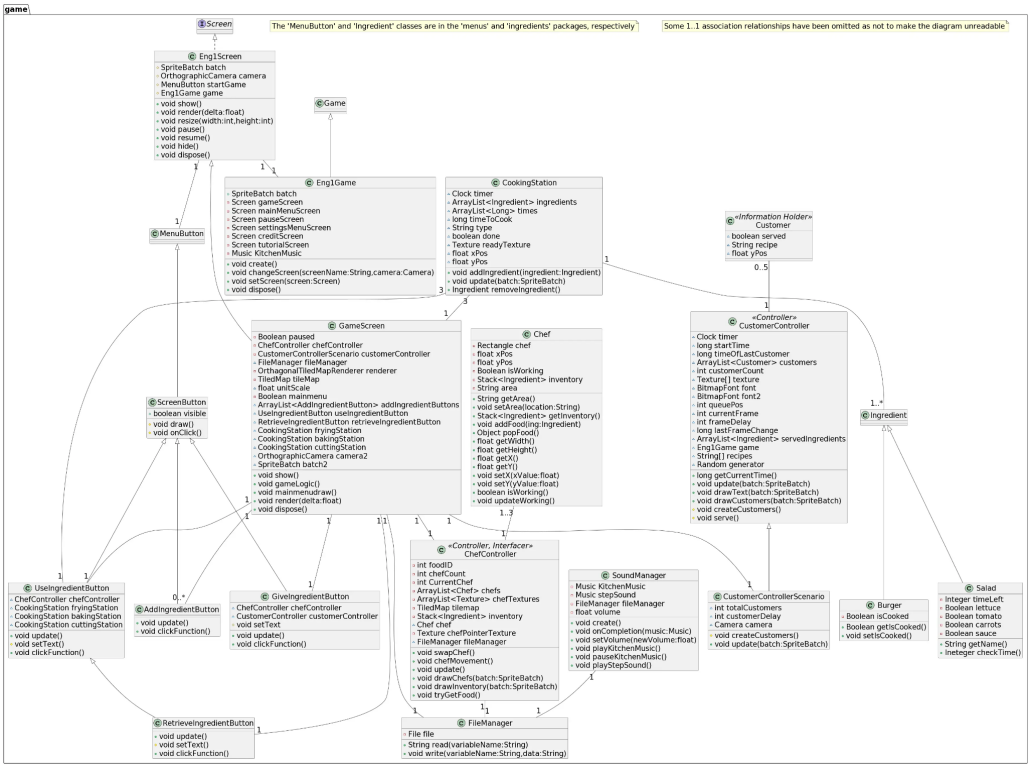
The architecture for our Piazza Panic game is displayed as UML class diagrams. To build these class diagrams, we used the PlantUML open-source tool and because we created these in Google Docs we used the PlantUML Gizmo extension/add-on tool. However, after encountering some issues with the extension, the class diagrams are now images with their corresponding PlantUML url links (copy and paste a link into a browser to access a diagram and it's UML). We chose PlantUML because it made the creation of the diagrams easier as the software uses a straightforward plain text language that was quick and easy to learn.



http://www.plantuml.com/plantuml/uml/rLPVQnmt47_Nfn3EGspS2xYq58vI4fiuDTGbz8BRF9MvtRbTSVgpI2F_YQa_UrUhNKlnAUipVIeStCnevgUHvZUZVUi8B7abcTiAtO45nMlSW9TqRZIjn1Vat__8M0VL5Xha1mgrTmVtZ6b3lReo2ld9cpSd79mpbG0IHdEBSX1SAph7MxX6laRKt2X1X7Nlo_jmJeOhs2Ad5daje0cRUsqHQbXBF0Xklvymzbut9wlA1e23amE-M0n_bwZzgISoch5B43heJxn-oBsAJX7g_QsW3w1mWllaoLW1SjVph0NTOBHFtYia4hfn1UoV-WYZoHeP0sF8J4JuEnSIBq53W_OvJo_772zDlH_ubP5UuKgAkaTd7y4xdC-Ww5A-YC9HluNAAWdEvQR8fxNfezGoN-3HLVG_FJ2b0OLyvEC4b2WISIHkQ9bPzm99NBbdhrPaXMuuuIsnLq8JI_reN9Jk7gKRKLCRnHP5q_Qkd_e0Rv53HU8Q_wQe9IiQNC5QIL5jiKuBgyuAmbEWgkr3gEftEGE55dWr_B5NfyP813s59eCIBja4AUX29_EDqNrwrqRKtEYpu71un8w37Y7ElQv2p_RMw5LRk7bqyLqDX46TKcwGHZ5On-kAnc0GceBsLoP_pkG_myrb-g_3zJrkvqBaDKYFozuXitukMIDQSINY5fPYzkD2pFw8_JDHOLMX1YlCbQZ8sBkHH7q0wZfdsprx-Bei4TtsRcWFQvG4ouqqG1bste8ZPEHUDDspHSx-JPBBrKh0qzXAF4xm6RKNKmKd_mlLi4KuKpy3OsSyjzCVwBp2kJrU19SEFv_m6cpgaE5QkHI2ha8LsN4qDKY3Ff4f9tBE_RH_cADkcgDpnve_Htf7zhwO9_sOeCuire8YoVQfPJNukvUNCX_a--Uz6FiztjgpU-GFmzmc8seL_NSRGKMPBTRZhBi1GNt4kR4NklDKJclqDaY3qyvClubEdmXp_PvrboEmGy6_jgO7U3JborEbyjKTtaIr44mxBPvtHxLjhF6wlb3XLLjCWowcVXWA5yNbXqVnQ6PYnHNNcjGzjN2TSLYQEkET2TL9LAZ9yUdAZdSVi-FY32fmKOqLYMeXfPdCBJhn1G-7HphC_k_Y6vq_0nruXncKyogfhQXnxkJfqwX8bLIPqoULUNZwBESbRanW8NkOUGJ-KxtoxyEPvBKXNF9tKlBra9VZWNwjS6FFSx38XUQln-NNd3syinRkVXEEVaxd-MMA_-Q9lIUqrodBy_Bb71mxydz5mVxt5dNjbVmN



http://www.plantuml.com/plantuml/uml/bP11Yzim48NI-XN2UsaRl9IUGrYIKchOfIK9lPQnF5P5PCb8ez0GqjzUOMBB2Kx2tloUtdppF2-Um55eD6j1x40YpvIH3Yk5XdnsP6oDGhNH_Ipy_2NZUTvzVSfuCSGO4ngyv-VaV5wOshe6l5d3lrjTeLiiZYn_LPxcQMp1HvjOlhHM8nYk_1gXEh1S6UAkGnPVsQbVakQFx2bk931IumZ5dhRubu933eBK7ej6ijc6N4nm0msoMNdULBj3D7zxwfecNmIYoOSITfD-xHHM5x4aErmIhRC2lSVutjUKIB7ZXuzijhUgwXDH3__yXa0tpieZ14rrTPSU7OomDfQ9_6GcAo-7cwywS-nyqwfjiJgtggEwL_4NIYGeDJui5EYoJQTlWrT_4FqqzT-8eHgPT32FgClW9BfbC0_1nVl06mXYiktLwfz0T1XoRmYZn2iI9ShejYdomoXhxWMsjW6ozn8hSCxITEB4Ng8P6lqV



https://www.plantuml.com/plantuml/uml/jLZVRzks4t_dNq4u3_7tcpXeCcmFHX6qSPSsMD8MSTPk6FR0K6UPYqHg9ENK3UQ_VKTI4YLBTWeK0ugOT_pmVlDuwYjZcRPbdX7p86J1DCjf0WjMPlPIIJiNNu6U_KH8mVW3Iu6EKfR3Y3uH8fL5SgvoeASd9wUK6QEuO5OeIJLaVc6MeZ1qoLP0xm4aLRcm5XB4KZnEhQ8vUm1gbq0JmLAdI9GQMCBkCnWvsFlJIj4TOWvlG9ONfRLA7b8c4tfu9LCDYG1f3odFq0Gmb6cWGbBh3kHum5HOqM2HK_bZZga6Km2tOWNPEgWy8sHM6ekUQJfwSNBouuXE9dHqEg8rUwQarIhBG2Dq2GiwmX3y4678seP46Y4Dyb9aSCCaIprCASn1EhSXY2CK652_6eYE1Iy7mG7zbkKmvngW3mgAdR7_7z4eaf1u9e9GabZr5Gz3Ry5g0TyCZnxjHRxmXhvEaltWNO4V1IMJEESWcHQgVw2Vf9zbUjg7EAuxl6CxrYqXSrNAf1AtKuklPU_E4vdS0hTCfXbGZWmoMMIAMVhbepBrUktN5qfbm2GLvhFIBkzaChTOvozZzCwfa2lyLNhjDZM2t7rXvAWYKh2laHx_Znojb4YegUXCSLz3qu33xRveF7jLIsy4i2IvL2ePuw5f1EFkX_k_W5jQgCBl8oVuWb8-YyGkEvnt8DAbxR1-wr2_HulTrfTFB2jXwhVZ1gBMtOtxxP0rOieYQHOYzvywAUtyON9kIYCu_KLOp8tqHCqp5WeF9fDMRkaYhklqhLHMvb0fvXgOXSOE9MSgBp9mmH_dJl3KYszvzSaV7qju3AkksqN6rctpniq6AmqCx-2HUMrzs_TGsQ6IANon81FJAkwd6D2LC09xE3dm0XFD7b7EGUrH9lX3Iswl7KJnKMTGiDdgfRn3udQtuotqEbAztNlsi6UNeyf8TigYySM0lKD3cyXnvzbbARaxqlOmjjBITZ7shmZfiVQQ-vonmyqy6ZoKhwVl2r1yjQAMs4kVOdFpLOjNQd-G9hkR_52mDZilw0TjbonLacLtoCPxNzoYRQ0HfyC2GVKMjVveGJRXNfTIs3bd6KJpSoQa6qh8vhNMR7qjZ7qvyBwTKzPd6h8PkYXbdqSsErEeTspybrTduwAeKisAfU0pXh5dbFkVCxAP5rfOk62MBr4g_ZshEzjIFJPraoByMgMnIJMHJ3IBi92JSGAPPTqsc0XJA0Fjacib_IdqQ9_xvYrqW-XunRHmq-XxD20-VbxaepDy272SCCjgSDHN3mbDWJi2xqqI0cHItOGqZd8roxdjEcwN7Qx55i-0pu8PW7ENUxl5jALLw4iMoA6dRVY9uaicqqhvsFYVTZ12fgSXxwrdgtDW6gWzsMfDLi8zYBaRv6dT7mxwGJpOMNNa8BvPr7y0-Tdje4cGYtkimdtb5NUC-0hZHpUvJ8Mqnqi_iRXbvxrjYJ9b7icbI2BHztxFqDfKKn-2RmdgfrPPbjMx9l1uHRw1ZAt9PYiyhImtvxOcex4FyhGLzcXjdDYVVUpQ8fz_92vAdO9kH4KTJ-tfrpKZICXHpN5pQSMCKsA75mSSbh5aMFwLjE2KMu6T21OsQip0sf93P630c5MHvanhPKra68RuzipzdeLQk98GjF0by8SxaGyM0Ns9S_T2wJnqstSgIq2VdsEwxsi5e5UGa4tJoRWeW6pYryA-oaBnaV1YPvdY3zvpJJQPaccuDSw-GFkz3ujhPcmjh_t2rRnpoeVBSAPAQSd6ZGgbXZ_-f3QioEP2s9mLbmh12_pJOvm5GN-NKC97rOZL6gIzr2o7m5cuPIZsO33w7lPdldkrhMr_quH0HWugw7I9EiHeSWWo0cwPJ5HEKv2WcLKuYsKgf7aMx0fvVVOw7JMpiuIBVaIyfdtOhliS-C75Tfd5qQfN3SVKNGEzO3nKniJ7LgdawarOXpaCand8oc4daKnYX1rzxh6cpQxI7tOgmcoDLvZxlP-rONoklfGh6Pxwg8GplMBj7Fdg_l382YSezYGaRjJAFplRkUiamZcrFrVn8tadpEhrMx3XmpWcfZjbTU_bhW8MxTADa6ibqtD_qhHky9roAfgsOTS5n9wIA0dD5VN3qRe9PlmlYNQJ2SvqHzbnD6WQkTzm6I8OTUUgFS7w6E6_Lo2JCi_-1G00

To view larger versions of our class diagrams, see our website:
https://decassociation.github.io/

We began designing the architecture by doing Responsibility Driven Design (RDD). We first came up with the themes for the game. Then we made CRC cards and on them we wrote the names of some candidate objects. Then on each card we wrote the purposes of the object and its role stereotypes. After that, we grouped the objects into customer, cook, menus, stations, food and general. We then removed objects which were too similar to another, and then on each card we wrote the responsibilities of the object. Once we were happy with the cards, this was all compiled into a Google Drive folder to make it easier to reference, and then from this we started to create the class diagrams in PlantUML. We made sure each class had the right attributes and methods to satisfy the responsibilities we had given it, as well as making small changes where it made sense. To view our CRC cards, see our group's website: https://decassociation.github.io

One significant aspect of our class diagrams is that they are not 100% made from our CRC cards. RDD and from that our CRC cards gave us a base for our game architecture however once we started actually developing our game and converting the CRC cards to class diagrams we realised that we were missing a lot; for example we didn't have every ingredient yet and we didn't know what would be best for certain features such as the timer - would this have it's own class or be embedded in another class? Some of the simplest things weren't set in stone. So, from our initial plan/base, the class diagrams have evolved a lot as we realised we were missing classes, functions and variables because we can't predict everything and often key details would arise during development of our game. Essentially our initial architecture wasn't entirely suitable so it changed a lot. To view interim versions of our class diagrams, see our website: https://decassociation.github.io/

A key and noticeable aspect of our class diagrams is that they are split into packages - 'game', 'ingredients' and 'menus'. Each package is a separate diagram because it makes the display of the classes easier as each package contains related classes (some classes are referred to in multiple diagrams because they may be interacted with by multiple classes in different packages, but each class is fully defined in one of the packages), as well as making it easier for multiple team members to be working on the diagrams simultaneously. All our classes and packages are not only based on our CRC cards from RDD but they're also based on the requirements for our Piazza Panic game. Our requirements, as they should, gave us a strong guideline because the requirements are of course paramount in our development; they act as a sort of checklist for us to always refer back to, to ensure we meet our customer's needs.

Our architecture meets all of our requirements for the game. The table below shows and explains how most classes work with or relate to one another to meet each requirement.

Classes	Requirement ID	Reasoning
CustomerController, CustomerControllerScenario, Customer, ScenarioVictoryScreen, GameScreen	UR_GAME_TIME, UR_TIMER, FR_TIMER, UR_SCENARIO_MODE	CustomerControllerScenario implements CustomerController (which interacts with Customer) and interacts with GameScreen to provide the scenario with 5 customers with a timer for the whole scenario. ScenarioVictoryScreen provides a victory screen for the scenario.

Eng1Game, PauseScreen, MainMenuScreen, CreditScreen, SettingsMenuScreen, TutorialScreen, MenuButton, Eng1Screen	UR_PAUSE, FR_PAUSE_SCREEN, UR_MAIN_MENU, FR_MAIN_MENU, UR_CREDITS, FR_CREDITS_SCREEN, UR_CONTROL_CHANGE, UR_TUTORIAL, FR_TUTORIAL	All required screens either extend MainMenuScreen or Eng1Screen, or implement Screen. All buttons on all of the screens extend MenuButton so that is central to navigation across all of the screens and meeting our requirements.
TutorialScreen, FileManager, SettingsMenuScreen	FR_TUTORIAL_CONTROL_CHANGE, UR_CONTROL_CHANGE	The tutorial screen doesn't show controls so it doesn't have to update it's controls. FileManager is used by SettingsMenuScreen to store the player's control settings, so if they are changed they're stored for next time the game is played.
SoundManager, GameScreen, MenuButton, MainMenuScreen, CreditScreen, TutorialScreen, PauseScreen SettingsMenuScreen	UR_SOUND, NFR_COLOUR_BLIND, UR_COLOUR_BLIND	SoundManager controls the music and sounds in the game. None of the screen classes have been tested to see if they are suitable for colour blind users but these screens are related to our colour blind requirements.
Chef, ChefController, CookingStation, CustomerController, CustomerControllerScenario, Burger, Salad, RetrieveIngredientButton, UseIngredientButton, AddIngredientButton, ScreenButton	UR_COOKS, UR_CARRY, FR_COOK_INTERACT, FR_MOVE_COOK, FR_DROP_INGREDIENTS, FR_SWITCH_COOK, UR_SERVE	The CookController class defines the controls and general logic for all of the cooks that the player can play as. The chefs can interact with cooking stations with the different buttons such as RetrieveIngredientButton (from an ingredient station) to make a Burger or Salad. Chefs can also interact with customers to give them their order with CustomerController and CustomerControllerScenario.
Customer, CustomerController, CustomerControllerScenario	UR_SCENARIO_MODE	CustomerControllerScenario controls how customers appear, implementing CustomerController and working with Customer to make organise the scenario.
Recipe, Burger, Salad, Ingredient	UR_RECIPES	The Ingredient class is the superclass for Burger and Salad, and all ingredients such as Carrot. Recipe uses a list of ingredients.
Ingredient, Vegetable, Meat, Sauce, Cheese, BurgerBun, Carrot, Lettuce, Tomatoes, Onion, BurgerPatty	UR_INGREDIENTS	Each ingredient has its own class and there is a hierarchy for type of ingredients as some are vegetables which are clearly not the same as a meat ingredient. All ingredients extend Ingredient.
CookingStation	UR_COOKING_STATIONS	The CookingStation class is the superclass of all the different cooking stations. Each type of station has its own, specific functions but they all have some of the same variables such as a list of the ingredients that they can use, cook, cut, etc.There would be a lot of repeated, redundant data if it wasn't a superclass.

CookingStation, UseIngredientButton, GameScreen, AddIngredientButton, RetrieveIngredientButton	NFR_QUICK_COOK, FR_COOKING_STATIONS, UR_COOKING_STATIONS	The cooking stations are implemented with CookingStation and GameScreen. There are all the required types of cooking station and chefs can interact with them all using buttons with pop up when a chef is standing at a cooking station. Recipes don't take long to cook at stations.
CookingStation, AddIngredientButton, Chef	FR_INGREDIENT_STATIONS, FR_UNLIMITED_INGREDIENT, UR_INGREDIENTS, UR_PANTRY, UR_RECIPES, UR_CARRY	Chef interacts with CookingStation through the AddIngredientButton whereby if the player is standing in the area of an ingredient station, the player can add an ingredient to the stack they are carrying by clicking an AddIngredientButton button. There's no limit on how many ingredients can be retrieved from an ingredient station. These ingredients are used to make recipes.
TutorialScreen, MainMenuScreen, MenuButton, FileManager, ControlChangeButton, SettingsMenuScreen, CookingStation	NFR_EASE_OF_USE	All aspects of the game will be designed with ease of use in mind, but these classes are most likely to have an effect on ease of use through clear and simple navigation. Each screen class inherits from Eng1Screen so they will all have a common, standard structure that will make the menus easier to use since they're consistent.