CALAMITY: A Disaster Preparedness Application

Software Architectural Design

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:

Dalisay, Nephia Bianca
Tan, Gene Audrey
Wee, Filbert Heinrich

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2019-2020

Unique Reference:

The documents are stored in the https://github.com/geneaudrey/Disaster-Preparedness/tree/master/03-Design%20Engineering referenced with Disaster Preparedness App - Architectural Design.pdf.

Purpose:

The purpose of this document is to show the revised software architecture model (interfaces, controllers, direct access objects, transfer objects, data sources) as well as the attributes and methods in each component.

Audience:

The following are part of the target audience:

- People who like or are interested in playing games
- People who have little to average knowledge on disaster preparedness, or people who would like to refresh their knowledge on disaster preparedness
- People who are interested in the development of the application

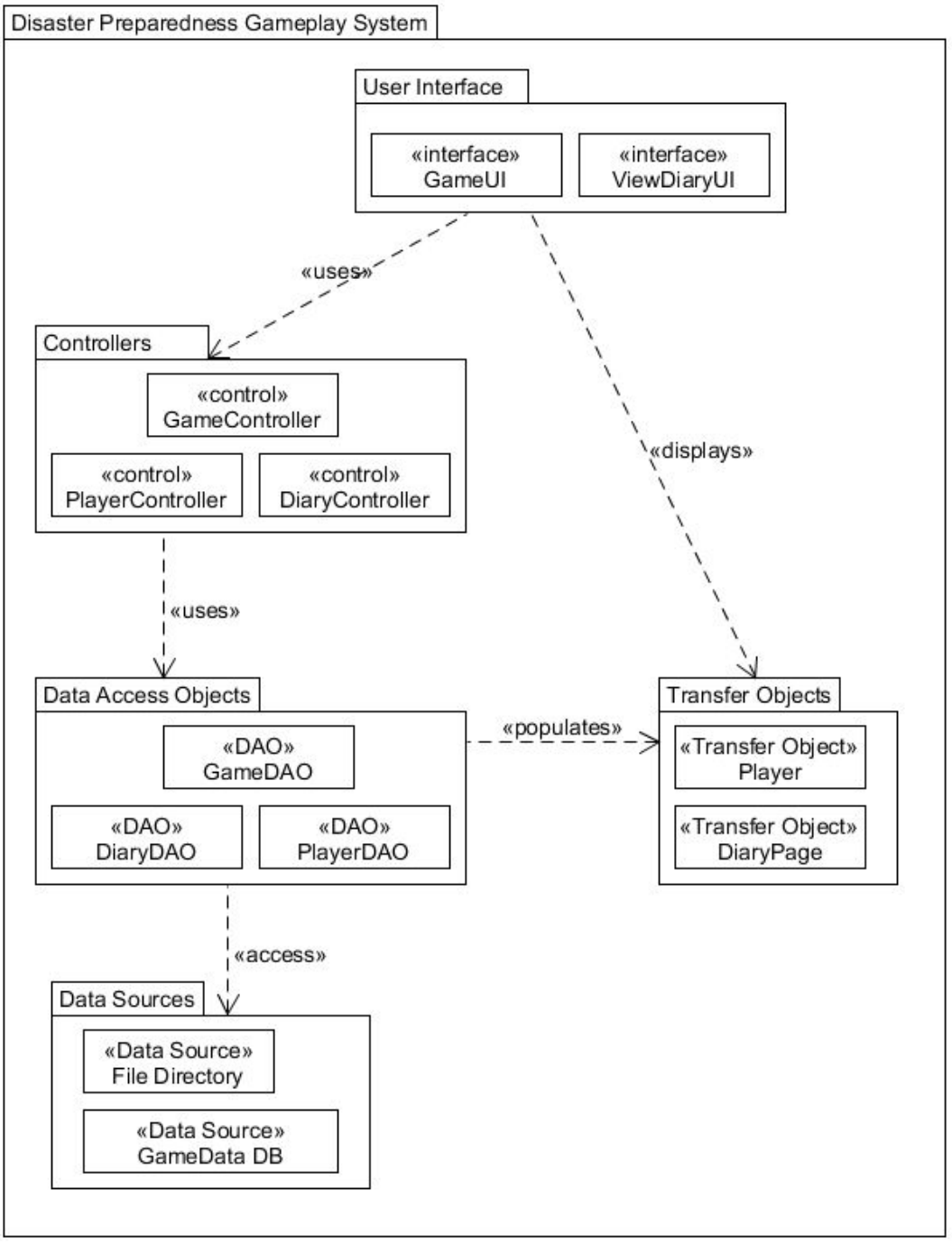
Revision Control:

Revision Date	*Person Responsible*	*Version Number*	*Contribution/Modification*
10/28/19	Gene Tan	1.0	Created Initial Document Added purpose and target audience of the document Added initial Revised Software Architecture Model and system name and description Added descriptions for data sources Added initial information on User interface packages added initial information on data access object packages added initial information on transfer object packages
10/30/19	Filbert Wee	2.0	Modified information on User Interface Packages Added information on Controller Packages Modified information on Data Access Object Packages Modified information on Transfer Object Packages

***System Name*:** Disaster Preparedness Gameplay System

Description: The system is an application that aims to encourage disaster preparedness in a fun, unique, and memorable way. With this, the central feature of the system will be a game. More specifically, it will be a game that teaches players what to do in certain moments of disaster (i.e: fires, earthquakes, floods, and storm surges) by forcing them to make choices within the game. Different combinations of decisions may lead to different endings. Additionally, a "diary" feature will also be included within the system. The pages in the diary are to act as "achievements" or bonuses that are unlocked when the players reach certain checkpoints in the game. The diary pages will be containing more information on the disasters featured in the game.

***Revised Software Architecture Model*:**



User Interface Package:

Screen Name	**Description**

GameUI	This is the screen of the player whenever he or she is playing the game (i.e.: making decisions or playing minigames in the game).
	Attributes:
	private Dialogue messages[]; //will contain the dialogues the player will be able to read in the game
	private SelectField choicesIDs[]; // when a player selects a choice, the choiceID selected will be saved
	private Choice selectedChoices[]; //will contain the 'replies' of the player in the game
	private Assets assets[];
	Responsibilities:
	private startGame(int scenarioID) : return Assets; //fetches assets for selected scenario
	private fetchDialogue(int choiceID) : return Dialogue; // the parameter contains the choice made by the player and the function fetches dialogue after player makes a choice

ViewDiaryUI	This is the screen of the player whenever he or she opens the diary.
	Attributes:
	private DiaryPage diarypages[]; //a list of diary pages
	Responsibilities:
	public isUnlocked(DiaryPage a) : return boolean;

Controllers Package:

Controller Name	**Description**

GameController (abstract)	This is the control that manages what the players can see/do in the game. It is considered an abstract class.
	Attributes:
	private Scenario scenario; //selected scenario
	private Dialogue dialoguesShown[]; //already shown dialogues
	private Choice choicesChosen[]; //already picked choices
	private Ending endingsUnlocked[]; //already finished endings
	private Assets assets[];
	Responsibilities:
	public fetchDialogue(int choiceID) : return Dialogue;
	private startGame(int scenarioID) : return Assets; //fetches assets for selected scenario

DiaryController (abstract)	This is the control that manages what the players can see/access in the diary. It is considered an abstract class.
	Attributes:
	private DiaryPage diaryPages[];
	Responsibilities:
	public unlockPage(int choiceID) : return void;
	public isUnlocked(DiaryPage a) : return boolean;

PlayerController (abstract)	This is the control that
	Attributes:
	private Player thePlayer;
	Responsibilities:
	private updatePlayerName(Player thePlayer, string name) : return void;
	private updatePlayerProgress(Player thePlayer, int dialogueID) : return void;

Data Access Objects Packages:

DAO Name	**Description**

GameDAO	This is the data access objects package GameDAO, which returns the possible gameplay flows for a scenario.
	Attributes:
	private Scenario theScenario;
	private Dialogue messages[];
	private Choice choices[];
	private Ending endings[];
	Methods:
	private startNewScenario(Scenario theScenario) : return Scenario;
	private getNextDialogue(Choice choice) : return Dialogue;
	private getChoices(Dialogue message) : return Choice; // the dialogueID will be from the Dialogue obtained from getNextDialogue
	private getEnding(Dialogue message) : return Ending;

DiaryDAO	This is the data access objects package DiaryDAO, which returns the contents and properties of a diary page.
	Attributes:
	private DiaryPage pages[];
	Methods:
	private unlockPage(DiaryPage page) : return void; //similar to updatePage()
	private getDiaryPages() : return DiaryPage[]; //returns a list of diary pages

PlayerDAO	This is the data access objects package PlayerDAO, which returns the properties and attributes that the player has unlocked during his/her gameplay..
	Attributes:
	private Player thePlayer;
	Methods:
	private updatePlayerName(Player thePlayer, string name) : return void;
	private updatePlayerProgress(Player thePlayer, int dialogueID) : return void;
	private getPlayerDetails() : return Player;

Transfer Objects Package:

Class Name	Description

DiaryPage	This is the transfer objects package Diary, which contains the data about the diary.
	Attributes:
	private int pageID
	private int choiceID
	private int scenarioID
	private string pageTitle
	private string pageContent
	private boolean unlocked
	Methods:
	private void unlockPage(int choiceID) : return void;
	
	private getpageID() : return int;
	private isUnlocked() : return boolean;

Player	This is the transfer objects package Player, which contains the data about the player.
	Attributes:
	private int playerID;
	private int playerName;
	private int dialogueID; // to store or save the current dialogue the player is on
	Methods:
	private updateProgress(int dialogueID) : return void;
	private updatePlayerName(string playerName) : return void;
	
	private getplayerID() : return int;
	private getDialogueID() : return int;
	private getplayerName() : return string;

Data Sources Package:

File Name or Database Name	Description
File Directory	This data source is a file directory that will contain the assets to be used in the game (music, images)
GameData DB	This data source is a database that will contain the data to be used in the diary and the game (dialogues, choices, player progress, etc).