Software Design Document (SDD)

Group 1: Travel Safe

SFWRENG 2XB3: Software Engineering Practice and Experience: Binding Theory to Practice

Version 1, April 12, 2020

Name	MacID	Email Address
Abeer Al-Yasiri	alyasira	alyasira@mcmaster.ca
Gary Gong	gongc12	gongc12@mcmaster.ca
Almen Ng	nga18	nga18@mcmaster.ca
Veerash Palanichamy	palanicv	palanicv@mcmaster.ca
David Yao	yaod9	yaod9@mcmaster.ca

Revision Page

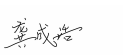
Revision History

Date	Version	Description	Author
April 12 2020	1.0	Initial release	David Yao

Roles and Responsibilities

Team Member	Student Number	Roles and Responsibilities
Abeer Al-Yasiri	400198787	Designer/Developer
Gary Gong	400182059	BackEnd worker
Almen Ng	400186180	Group Leader, Front End Leader
Veerash Palanichamy	400196473	Back-end leader
David Yao	400207967	Back-end, documentation

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through [CS] or [SE]-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.

Name	Signature
Abeer Al-Yasiri	
Gary Gong	
Almen Ng	
Veerash Palanichamy	

David Yao

Contribution Page

Name	Role(s)	Contributions	Comments
Abeer Al-Yasiri	Designer/Developer	-Assisted when necessary with backend and frontend development - Writing documentation	None
Gary Gong	Back End worker	-Connects Google map API, adding permissions in XML file. -Adding messages to notify connections on API and Internet - Writing documentation	None
Almen Ng	Group Leader Front End Leader	- Designed the Interface - Worked on front end (with creation of buttons, spinners, markers, etc) - Commented on code - Writing documentation	None
Veerash Palanichamy	Back-end leader	- Laid out the class interactions and which algorithms would be used - Created UML	None

		diagram - Implemented graph algorithms	
David Yao	Back-end, documentation	- Helped design icons for markers - Improvements to visual appearance of final app - Writing documentation and summaries - Built classes to interact with datasets - Implemented merge sort and graph edges	None

Executive Summary

The project tackles a niche problem that individuals face during their travels. We usually dress accordingly to how the weather is; but it is more important to keep up with any weather risks that may occur within range of your location/destination. This valuable information will help in determining the most suitable location for leisure or business trips. Our team focuses on building a software application using the US National Oceanic and Atmospheric Administration dataset composed of severe weather events across the country to map, organize, and rank any extreme and dangerous weather conditions that deem to be unsafe or undesirable travel destinations. The program's analysis will rely on analyzing weather activity history of a specific geographic location and using longitude and latitude coordinates. Therefore, this will allow for a complete implementation of the program to help the client with making business and travel decisions.

Table of Contents

1. Overview

1.1 Scope

Travel Safe aims to help with travel decisions with the aid of weather predicting software application. The objective is to use a detailed database to draw relations between the input statistics that will deliver a system to assist the user with choosing their future travel location. The system will offer a user-friendly Android based application that allows users to search a specific location's weather status. The application will provide information regarding the occurrence probability of extreme and unwanted weather conditions. The product's main goal is providing the user with beforehand information to aid in the traveling decisions that are affected by unsettling weather changes. The project's version one implementation will be done in a time frame between March and April 2020.

1.2 Purpose

This document outlines a description of the product's design to allow for software development to proceed by providing detailed structural interactions of the design systems and modules.

1.3 Intended audience

The intended audience of the software design descriptions document is developers who are interested in implementation and test; as well as technical stakeholders.

1.4 References

IEEE Computer Society. (2009). *IEEE STD 1016-2009* [PDF file]. Retrieved from http://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf

National Centers for Environmental Information. (n.d.). *Storm Events Database.* Retrieved from ftp://ftp.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/

Gregg Lind. (2009). *Adjacency List of States of the United States (US).* Retrieved from *https://writeonly.wordpress.com/2009/03/20/adjacency-list-of-states-of-the-united-states-us/*

2. Definitions, Acronyms and Abbreviations

Android: A mobile operating system (OS) developed by Google

Database: An organized collection of data.

Eclipse IDE: An Integrated Development Environment (IDE) that enables many customizations and extensions. It is known for its Java IDE.

IEEE (Institute of Electrical and Electronics Engineers): A professional electrical and electronics engineering association dedicated to technological innovation and advancement.

Graphical User Interface (GUI): The visual output displayed by an application to the user to aid user interaction.

Software Design Document (SDD): A document that represents a software design to be used for communicating design information to its stakeholders.

User: A person who uses the software.

Stakeholders: Any person with an interest in the system who is not a developer of the system.

Unified Modeling Language (UML): a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems.

Class Diagram: a central modeling technique that runs through nearly all object-oriented methods.

Use Case Diagram: a model of the system's intended functionality (use cases) and its environment (actors). Use cases define relation from what a system to how the system delivers on those needs.

3. Conceptual Model for Software Design Descriptions

3.1 Scope

The final product is a mobile application that is capable of displaying all natural disasters occuring in 2018 and most of 2019 for any given state in the United States. The main interface takes the form of a map, with markers corresponding to each disaster occurrence over the two years in the given state. Users are able to tap on individual markers to view information regarding that specific disaster.

3.2 Purpose

The purpose of this document is to fully describe the software design of the final application, Travel Safe, in a way as to highlight the software's structure and development process. It is intended for use by the product's designers and maintainers to allow for future updates and improvements as is necessary. The document will discuss structures, their interactions, and the resulting outputs as per the requirements laid out in the relevant document.

3.2.1 Influences on SDD Preparation

The influence on the software design process is the SRS document. Due to the functional and non-functional requirements specified earlier in the process the developer has a basis to plan the SDD.

3.2.2 Influences on Software Life Cycle Products

Throughout the design process, some features of the project have been modified and improved upon to better enforce the requirements and constraints described in the SRS report. The changes made to the requirements have been done to better communicate the functions of the projects and that will be reflected upon in the test plans.

3.2.3 Design Verification and Design Role in Validation

Testing phase of the development process will show whether the specified requirements have been fulfilled. The functional and nonfunctional requirements are modeled in the design view parts of the document. The SDD is to be used to drive test cases to prove the verification and validation of the design models.

4. Design Description Information Content

4.1 Introduction

The design description of Travel Safe will outline how the system would be designed and implemented. This section covers the SDD identification, the design stakeholders and concerns, viewpoints, design views, viewpoints, elements overlays, rational, and languages.

4.2 SDD Identification

Travel Safe will be released on April 12, 2020 after proper testing has been done. Prototype was to be shown to the stakeholders outlined in 4.3 late March 2020. TravelSafe is ready to be released.

Scope, references, context, and summary are found under section 1, “Overview.” The Glossary is found in section 2, “Definitions, Acronyms and Abbreviations.”

4.3 Design Stakeholders and Their Concerns

Design stakeholders include Travel Safe developing team as well as interested personnels in the design development process and progress of the projects (such as project reviewers and other interested programmers) . The main concerns of the stakeholders include delivering a working implementation within the deadline date along with ensuring the usability of the app and how informative the information outputted is.

4.4 Design Views

Design views allow the design stakeholders to gain insight on design details from different perspectives and meet the requirements relevant to the perspectives. In order for the SDD to be complete, each design view must conform to the design viewpoint. It is considered consistent if there are no conflicts between the elements and the views.

4.5 Design Viewpoints

Context Viewpoint

A description of the relationships, dependencies, and interactions between the application and its environments. This viewpoint is suitable for Travel Safe as the interactions between interested personnels is important.

Composition Viewpoint

The composition viewpoint describes the way the program is split into components and the role each of these components have.

Logical Viewpoint

A description of the class structure and the implementation and interactions of individual classes. This section includes the class diagram, defining classes and their relationships.

Dependency Viewpoint

The dependency viewpoint describes the program's components and their interdependencies. It highlights information that is passed between components and the actions they take with the information they receive.

Information Viewpoint

A description of the data types in the program.

Pattern Use Viewpoint

A description of the design patterns used in the design process of this project.

Interface Viewpoint

The interface viewpoint describes the program's internal and external interfaces. Internal interfaces are between components of the program and involve the passing of information and commands. External interfaces describe interactions between the program, interface device, and the user via program output and user input.

Interaction Viewpoint

The interaction viewpoint describes the used relations and strategies among the modules in order to form a sequential behaviour of the program. The section will go in depth onto explaining the how, why, where, and what level will the specific actions take place in the program.

State Dynamics Viewpoint

A description of the behaviour of the program in terms of states and state transitions in response to data inputs. This viewpoint includes a state diagram defining the behaviour of the program.

4.6 Design Elements

Information on design elements include design entity, design relationship, design author, design attribute, name attribute, type attributes, purpose attribute, and design constraints is presented in section 5.0, Design Viewpoints.

4.7 Design Overlays

Information on design overlays is presented in section 5.0, Design Viewpoints.

4.8 Design Rationale

The application was divided into several classes in the application of object-oriented programming. The separated classes make interactions between the software datasets, parsers, and output devices clear by encapsulating specific functions in their own subprograms. Individual classes were made for sorting, data representation in objects and graphs, graph traversal, and application updates and output.

4.9 Design Languages

The Unified Modelling Language (UML) will be used in this document for the included diagrams, to provide a clear image of the structure of each component of the system.

5. Design Viewpoints

5.1 Introduction

The Design Viewpoints section highlights the design concerns and interactions corresponding to each design viewpoint considered in the development of Travel Safe.

5.2 Context Viewpoint

The context viewpoint describes the functions of a design element and its interaction with the wider system and the user.

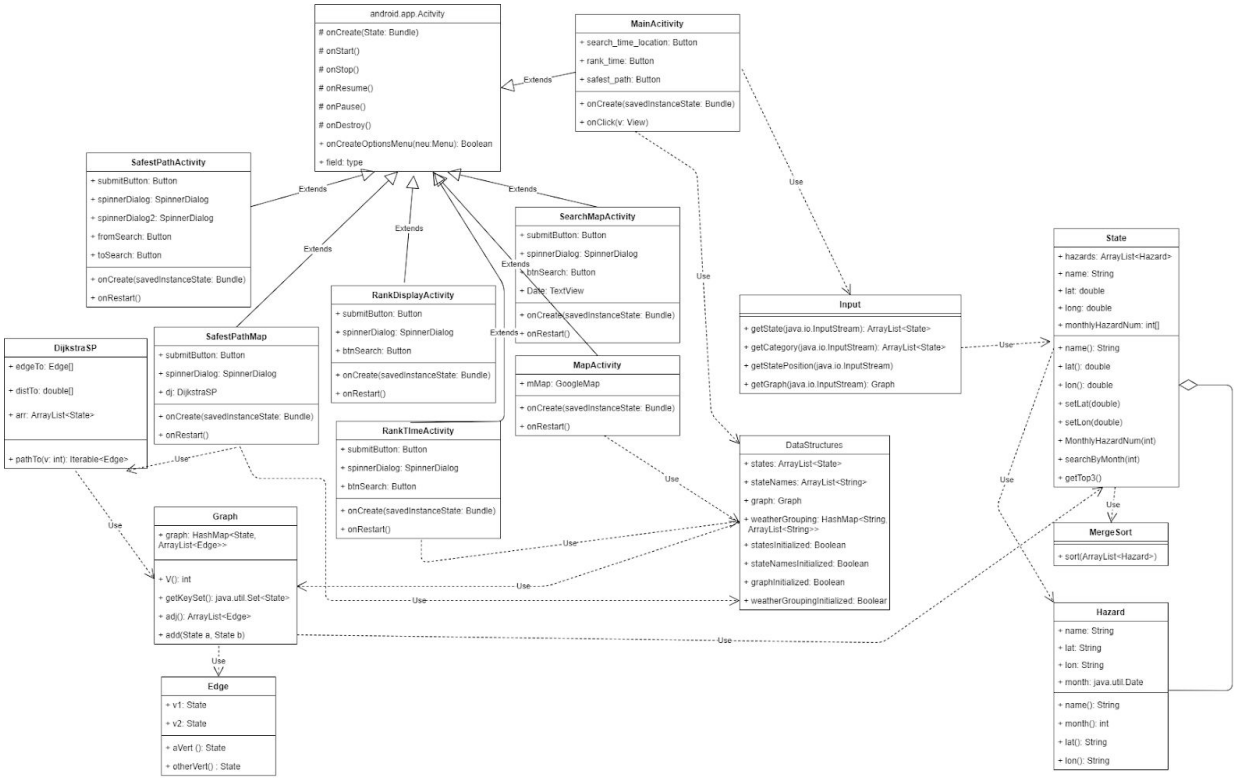
5.3 Composition Viewpoint

5.4 Logical Viewpoint

5.4.1. Design Concerns

This viewpoint highlights the implementation and use of abstract classes in order to fulfil the required function of the wider program. The utility of classes is highlighted in earlier sections - they simplify interactions between different segments of code to the programmer.

5.4.2. Design Elements



5.4.2.1. Data Structures Class

This class declares all the public static variables used throughout the entire project.

Description:

Name	Type/Return Value Type	Visibility	Definition
states	ArrayList<State>	public	ArrayList of State Objects
stateNames	ArrayList<String>	public	ArrayList of State Names (Strings)
graph	Graph	public	Graph of the state objects
weatherGrouping	HashMap<String, ArrayList<String>>	public	HashMap with strings as keys and ArrayList of strings as values to organize the weather events with their categories
stateInitialized	static boolean	public	Keeps track of if state has been initialized
stateNamesInitialized	boolean	public	Keeps track of if stateNames has been initialized
graphInitialized	boolean	public	Keeps track of if graph has been initialized
weatherGroupingInitialized	boolean	public	Keeps track of if weatherGrouping has been initialized

5.4.2.2. Dijkstra Class

This class is used to find the shortest path between states using Dijkstra's algorithm. It will use the frequency of hazards within a specified state and month as weights.

Description:

Name	Type/Return Value	Visibility	Definition

	Type		
edgeTo	Edge	private	Array that tracks the edges
distTo	double	private	Array that tracks of distances from source
pq	IndexMinPQ<Double>	private	Priority Queue
arr	ArrayList<State>	private	ArrayList of states
DijkstraSP()	DijkstraSP	public	Constructor of the DijkstraSP using priority queues
relax()	void	private	Relaxes edges
distTo()	double	public	Returns the distance from source to vertex
pathTo()	Iterable<Edge>	public	Returns the path from source to vertex

5.4.2.3. Edge Class

This class represents an edge on the graph that will be traversed.

Description:

Name	Type/Return Value Type	Visibility	Definition
Edge	Edge	public	Constructor for Edge, taking in two states as parameters.
v1	final State	private	Instance variable for the first State
v2	final State	private	Instance variable for the second State

aVert	State	public	Method returning the State v1.
otherVert	State	public	Method returning whichever State was not given as a parameter.
toString	String	public	Method returning a String containing information about the Edge.

5.4.2.4. Graph Class

This class represents a graph that can be traversed.

Diagram:

Description:

Name	Type/Return Value Type	Visibility	Definition
Graph	Graph	public	Constructor for the graph, initializes instance variable graph as an empty HashMap.
graph	HashMap<State, ArrayList<Edge>>	private	Instance variable for the graph, HashMap linking a State to an ArrayList of Edges. Adjacency list representation.
V	int	public	Method that returns the size of the graph.
getKeySet	Set<State>	public	Method that returns the Set of States in the graph.
add	void	public	Method to add an edge between two State.
adj	ArrayList<Edge>	public	Method that returns all the adjacent States to the State given as a parameter.
toString	String	public	Method returning a String containing the States and their adjacent States.

5.4.2.5. Hazard Class

This class represents an environmental hazard to be displayed on the map.

Diagram:

Description:

Name	Type/Return Value Type	Visibility	Definition
Hazard	Hazard	public	Constructor for the hazard, taking parameters for name, magnitude, year, month, day, lat, and lon.
name	final String	private	Instance variable for the name of the hazard.
magnitude	final String	private	Instance variable for the hazard's magnitude.
month	final String	private	Instance variable for the month the hazard took place.
year	final String	private	Instance variable for the year the hazard took place.
day	final String	private	Instance variable for the day the hazard took place.
lat	final String	private	Instance variable for the latitude where the hazard happened.
lon	final String	private	Instance variable for the longitude where the hazard happened.

name()	String	public	Method returning the name.
magnitude()	String	public	Method returning the magnitude.
month()	int	public	Method returning the month.
year()	String	public	Method returning the year.
day()	String	public	Method returning the day.
lat()	String	public	Method returning the latitude.
lon()	String	public	Method returning the longitude.
compareTo	int	public	Implementing Comparable interface, comparing the month the hazard occurred.

5.4.2.6. IndexMinPQ

This class defines the index priority queue object.

Description:

Name	Type/Return Value Type	Visibility	Definition
maxN	int	private	Max number of elements
n	int	private	Number of elements on PQ
pq	int[]	private	Binary heap array
qp	int[]	private	Inverse of pq

keys	Key[]	private	Array holding keys
IndexMinPq()	None	public	constructor
isEmpty()	Boolean	public	Returns true if priority queue is empty
contains()	Boolean	public	Checks whether priority queue contains specific index
size()	int	public	Returns size of PQ
insert()	void	public	Inserts a key at specific index
delMin()	int	public	Remove minimum key and return
changeKey()	int	public	Changes key at specific index
validateIndex	void	private	Checks whether an index is valid
exch	Boolean	private	Exchanges 2 array elements
swim	void	private	Swim function for PQ
sink	void	private	Sink function for PQ
iterator	Iterator<Integer>	public	Returns iterator of PQ

5.4.2.7. Input Class

This class reads and organizes all the datasets and text files.

Description:

Name	Type/Return Value Type	Visibility	Definition
getState()	ArrayList<State>	public	Reads the main data set of the weather events and organizes

			the data based on states
getCategory()	HashMap<String, ArrayList<String>>	public	Reads events_sorted.txt and organize data based on the categories as keys of the HashMap
getStatePosition()	void	public	Sets the longitude and latitude of state objects
getGraph()	Graph	public	Reads state_adjacency.csv and creates a Graph object

5.4.2.8. MainActivity Class

This class displays the app's main title screen to the mobile device.

Description:

Name	Type/Return Value Type	Visibility	Definition
ERROR_DIALOG_REQUEST	static final int	private	The error dialog.
search_time_location	Button	package	Button to search by time and location.
rank_time	Button	package	Button to rank the best three months for a given location.
safest_path	Button	package	Button to find the safest path.
onCreate()	void	protected	Method to initialize all instance variables when the screen is created.
onClick()	void	public	Method to do an

			activity depending on the button pressed.

5.4.2.9. MapActivity Class

This class builds a map and draws markers based on the SearchMapActivity class input.

Description:

Name	Type/Return Value Type	Visibility	Definition
bounds	LatLngBounds	private	LatLngBounds is object used by google map api to get boundary of markers
builder	LatLngBounds.Builder	private	Builder used to process and build the bounds.
mMap	GoogleMap	private	Google map object from the google map api
current_month	int	private	Field representing the current month for which data is being displayed
current_state	int	private	Field representing the current state for which data is being displayed
idmap	HashMap<String, Integer>	private	Data structure used to map hazard to their respective picture id
onCreate()	void	protected	Function triggered when activity is created
loadIdMap()	void	public	Inialise the idmap data structure with valid data

loadMap()	void	public	Load google map
drawMarker()	void	private	Draw markers on the map
onMapReady()	void	public	Function triggered when map is ready
onCreateOptionsMenu()	boolean	public	When menu bar is created
onOptionsItemSelected()	boolean	public	When item from menu bar is selected

5.4.2.19. MergeSort Class

This class runs merge sort on an ArrayList of Comparable objects.

Description:

Name	Type/Return Value Type	Visibility	Definition
aux	Hazard[]	private static	Auxiliary array to temporarily store values to be merged.
sort()	void	public static	To be called externally - sort the parameter ArrayList<Hazard>, using parameters for lo, mid, and hi indices.
sort()	void	private static	Called by public sort(), breaks down the array.
merge()	void	public static	Merges the subarrays back together.

5.4.2.11. RankDisplayActivity Class

This class displays the app's ranking screen to the mobile device, created when the user requests the top three months for the given location.

Description:

Name	Type/Return Value Type	Visibility	Definition
choice1	TextView	private	Display the first month.
choice2	TextView	private	Display the second month.
choice3	TextView	private	Display the third month.
onCreate()	void	protected	Method to output and display the top three months when the screen is created.
onCreateOptionsMenu()	boolean	public	Method to create a menu in the top right corner for options. Returns true when finished.
onOptionsItemSelected()	boolean	public	Method to return to the main screen by pushing the specified button. Returns true when it is pressed.

5.4.2.12. RankTimeActivity Class

This class displays a user input field to the mobile device, created when the user requests the top three months for a given location. Gets the location requested.

Description:

Name	Type/Return Value	Visibility	Definition

	Type		
state	int	private	The state the user requests.
submitButton	Button	package	The button to submit the request.
spinnerDialog	SpinnerDialog	package	The spinner to select the state.
btnSearch	Button	package	The button to search.
onCreate()	void	protected	Method to initialize this dialog when it is requested by the user.
onRestart()	void	protected	Called when this activity is finished and the app can return to the main screen.
loadSpinnerDialog()	void	public	Initialize a Spinner to request the user's desired state.

5.4.2.13. SafestPathActivity Class

This class displays a user input field to the mobile device, created when the user requests the safest path to a location from a location.

Description:

Name	Type/Return Value Type	Visibility	Definition
choice1	TextView	private	Display the first month.
choice2	TextView	private	Display the second month.
choice3	TextView	private	Display the third month.

onCreate()	void	protected	Method to output and display the top three months when the screen is created.
onCreateOptionsMenu()	boolean	public	Method to create a menu in the top right corner for options. Returns true when finished.
onOptionsItemSelected()	boolean	public	Method to return to the main screen by pushing the specified button. Returns true when it is pressed.

5.4.2.14. SafestPathMap Class

This class displays the safest path and the map being displayed on the mobile device after the user inputs their request.

Description:

Name	Type/Return Value Type	Visibility	Definition
bounds	LagLngBounds	private	LatLngBounds is object used by google map api to get boundary of markers
builder	LatLngBounds	private	Builder used to process and build the bounds.
mMap	GoogleMap	private	The map to be displayed.
current_month	int	private	The current month.
source_state	int	private	The state from which to search.
target_state	int	private	The destination

			search.
dj	DijkstraSP	private	To run Dijkstra's algorithm on the graph.
onCreate()	void	protected	Initialize the map display to the screen.
loadMap()	void	public	Load the map and the layout of the page.
onMapReady()	void	public	Main Map function - draws markers for hazards, animates camera, draws safest path
onCreateOptionsMenu()	boolean	public	Creates a menu in the top right corner. Returns true when complete.
onOptionsItemSelected()	boolean	public	Method to return to the main screen by pushing the specified button. Returns true when it is pressed.

5.4.2.15. Search Class

This class defines the methods to run Binary Search on a sequence of Hazard Objects

Description:

Name	Type/Return Value Type	Visibility	Definition
search()	ArrayList<Hazard>	public	Constructor for Binary Search
binarySearch()	Int	public	Binary Search function

5.4.2.16. SearchMapActivity Class

This class displays a user input field to the mobile device, created when the user requests to see the hazards of a specific month of a certain state.

Description:

Name	Type/Return Value Type	Visibility	Definition
current_state	int	private	The state the user requests.
current_month	int	private	The current month requested.
submitButton	Button	package	The button to submit the request.
spinnerDialog	SpinnerDialog	package	The spinner to select the state.
btnSearch	Button	package	The button to search.
Date	TextView	package	Display for the date.
onDateSetListener	DatePickerDialog.On DateSetListener	package	Set a listener on the date.
calendar	Calendar	package	A calendar to pick a date from.
year	final int	package	The year.
month	final int	package	The month.
day	final int	package	The day.
onCreate()	void	protected	Method to initialize this dialog when it is requested by the user.
onRestart()	void	protected	Called when this activity is finished and the app can return to the main screen.

loadSpinnerDialog()	void	public	Initialize a Spinner to request the user's desired state.
loadDateSpinner()	void	public	Initialize a Spinner to request the user's desired date.

5.4.2.17. State Class

This class represents US states.

Description:

Name	Type/Return Value Type	Visibility	Definition
name	String	private	Name of the state
hazards	ArrayList<Hazard>	private	ArrayList of Hazard Objects in the state
dataSorted	boolean	private	Tracks if the hazards have been sorted
monthlyNazardNum	int[]	private	Number of hazards in a month in the state
lat	double	private	Latitude of state
lon	double	private	Longitude of state
State()	State	public	Constructor for State Object
name()	String	public	Returns the name of the state
setLat()	void	public	Setter for latitude of state
setLon()	void	public	Setter for longitude of state

lat()	double	public	Getter for latitude of state
lon()	double	public	Getter for longitude of state
addHazard()	void	public	Adds a Hazard object to instance variable hazards and increments monthlyHazardNum by one
MonthlyHazardNum()	int	public	Returns the number of hazards in a month in the state
equals()	boolean	public	Checks if an object is equivalent to the current object
hashCode()	int	public	Returns the hash code
toString()	String	public	Returns the name of the state as a string
searchByMonth	ArrayList<Hazard>	public	Returns the hazards in the state in a specific month as an ArrayList of Hazard objects
getTop3	ArrayList<Integer>	public	Finds the top 3 months when the hazard frequency is the lowest.

5.5 Dependency Viewpoint

In the project the system that was deployed breaks down to three components that represent specific aspects of the program: Data management, Business processing, and Presentations components. The data management component uses external data sets to form the application

database server that will be the mainframe of the systems. The business processing component encapsulates the logical development process and the domain concept. Lastly the presentation component will focus on delivering the user interface platform to display the client application.

The project's user interface is covered under the *app/main/res.package.* The features are implemented using Android technology for client application to represent the modeled data. This facilitates the usage of the program by being controlled by the user's inputs and interactions.

The business component consists of the transition modules that includes the processes commands, logical decisions and performances. The packages responsible for that include: *src/MyApplication/app/src/main/java/com/example/myapplication.packages.* This package has a direct relationship with the user interface as it is the backend development of the TravelSafe app.

Tha data management component is responsible for use for storing and retrieving the information from the database. This information will be passed to the Business packages for processing and then eventually back to the user. The package responsible for the following actions is *data.package*.

5.6 Information Viewpoint

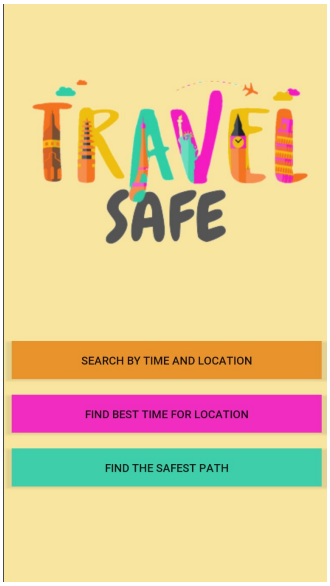
The project uses a weather conditions database which is primarily used to display the weather conditions on the map user interfaces. The main design concern related was preprocessing the data in time and space efficient method in order to be used for the upcoming development stage. The design elements related to this part have restricted access and it must be encapsulated by the design development. The database consists of information on specific US weather conditions as well as longitude and latitude coordinates.

5.7 Pattern Use Viewpoint

This project was developed using the adapter design pattern. The interfaces for mobile app output and Java program output are not typically the same, so classes existed to adapt the output from classes to input compatible with the API specified by Google.

5.8 Interface Viewpoint

The application's user face is controlled from the back-end by Java classes created for each distinct screen reachable from the main screen.



The main screen of Travel Safe

The main screen, for example, is generated from MainActivity.class. The user has three actions they can take - searching by time and location, finding the best time for a location, and finding the safest path to a location.

The interface between the Java code and the app was made using classes such as the one used to display the main screen.

The user interface was laid out using Google's Android Studio software.

5.9 Algorithm Viewpoint

Various algorithms were used. The main ones are merge sort to sort hazards by month so that the search data can be used for binary search to find the specific hazards occuring at a month. Graph algorithms and structure were used to find the safest path between states. Dijkstra was used while the edge weights were the hazard frequency for each state.

Critique of Design

- Modularity and separation of concerns were apparent in our implementation as all our modules had a specific purpose and there was not a lot of coupling. For example, we had a specific module, called input.java., that does all the reading of the files (.txt, .csv)
- There was a lack of consistency with class namings and variable naming.
- There was not much generality implemented, for example the merge sort only works for a specific type of arraylist.