Software Requirements Definition

CSCE 4444

What's There?

Deliverable II: Requirements Specification

Version 1.0 approved

Prepared by Peter Menchu, Garrett Brumley, Nasser Alqudaihi, Joseph Vo

University of North Texas

10/6/2017

Table of Contents

PART I: REQUIREMENTS SPECIFICATION

1. INTRODUCTION

1.1 PURPOSE

This document specifies the requirements for the What's There? web application, which provides city information for travelers and residents. First release. This document covers specifications for the entire product.

1.2 AUDIENCE AND READING SUGGESTIONS

This document is intended for use by developers, project managers, and testers of the What's There? web app. The suggested reading for this document is to follow the ordering of the table of contents. Introduction and overall description sections should be read first to gain an understanding of the app concept followed by functional and nonfunctional requirements to understand specific functionality. The interface section should be read next to understand the user interface and the presentation of the app, followed by the UML diagrams to understand how the functions fit together. The order of these two could be switched if desired. Test cases are to be read next to understand how the described app should be tested. The remaining sections cover our progress on the project.

1.3 PROJECT SCOPE

What's There? is a web app which is proposed to act as a central information hub for a selected city. The app will provide weather data, traffic data, hotel and restaurant prices, transportation data and links, and hardcoded geography/environmental info. The intention of the app is to provide travelers or residents of a city with a central place to find a bunch of information about a city. This would reduce the amount of searching to find the information needed by only requiring one search to find the city, returning an assortment of data. This would especially be useful for seniors, who may not be as adapted to technology and dislike having to search the internet for data.

1.4 PROJECT PERSPECTIVE

The concept of this app originated as a project idea for CSCE 4444, Software Engineering. It is a new and self-contained product.

1.5 REFERENCES

Coding

Video

[1] Traversy Media, "Web Development In 2017 - A Practical Guide," YouTube, 20-Dec-2016. [Online]. Available: https://www.youtube.com/watch?v=9hDKfBKuXjI&t=1s. [Accessed: 11-Oct-2017].

[2] Traversy Media, “Build An HTML5 Website With A Responsive Layout,” YouTube, 25-Dec-2016. [Online]. Available: https://www.youtube.com/watch?v=Wm6CUkswsNw. [Accessed: 11-Oct-2017].

[3] LearnWebCode, “JSON and AJAX Tutorial: With Real Examples,” YouTube, 26-Oct-2016. [Online]. Available: https://www.youtube.com/watch?v=rJesac0_Ftw. [Accessed: 11-Oct-2017].

Articles

[4] C. Cairns and D. Somerfield, “The Basics of Web Application Security,” martinfowler.com, 05 Jan- 2017. [Online]. Available: https://martinfowler.com/articles/web-security-basics.html. [Accessed: 11-Oct-2017].

[5] C. House, “A Complete Guide to Grid,” CSS-Tricks, 13-Sep-2017. [Online]. Available: https://css tricks.com/snippets/css/complete-guide-grid. [Accessed: 11-Oct-2017].

[6] O. Emmanuel, “How to Efficiently Master the CSS Grid in a Jiffy – Flexbox and Grid – Medium,” Medium, 07-Jul-2017. [Online]. Available: https://medium.com/flexbox-and-grids/how-to-efficiently-master-the-css-grid-in-a-jiffy-585d0c213577. [Accessed: 11-Oct-2017].

[7] ,"How TO - Slideshow," How To Create a Slideshow. [Online]. Available: https://www.w3schools.com/howto/howto_js_slideshow.asp. [Accessed: Sept. 11, 2017].

[8] "HTML Links,".[Online]. Available: https://www.w3schools.com/html/html_links.asp. [Accessed: Oct. 11, 2017].

[9] Laurence Bradford, "WEB, DESKTOP, MOBILE, OR CROSS-PLATFORM: OPTIONS FOR APP DEVELOPERS," Learn to Code With Me, Jan. 12, 2017. [Online]. Available: https://learntocodewith.me/posts/cross-platform-apps/. [Accessed: Oct. 11, 2017].

Weather

- Open Weather Map API

  https://openweathermap.org/

- Google

  Google API’s

  Street View:

  https://developers.google.com/maps/documentation/streetview/intro

  Directions API:

  https://developers.google.com/maps/documentation/directions/start

  Places:

  https://developers.google.com/places/web-service/

Maps:

https://developers.google.com/maps/web-services/

Traffic:

https://developers.google.com/maps/documentation/javascript/examples/layer-traffic

Deliverable

Book

[1] R. Pressman and B. Maxim, Software Engineering: A Practitioner's Approach, 8th Edition. New York, Ny: McGraw-Hill, 2015.

Digital Files

[2] U. Subedi, G. Verma, P. Amarakeerthi, and V. K. Arachchillage, “example-DELIVERABLEII.” .

[3] “good-example-DeliveranleII.”.

[4] “srs_template.doc.”.

2. OVERALL DESCRIPTION

2.1 PRODUCT FEATURES

System will provide a city search feature on the main page, and an account creation and login system that can save a home city and favorite cities for many different users. Software provides accurate and up to date information on weather, traffic, hotel prices, restaurant prices, public transport, news, and environmental for a given city. Software includes collapsible navigation sidebar on left hand side of all pages containing navigation to login and account services, cities list, the trip calculator, notifications, about and contact. The trip calculator will provide a list and total cost of expenditures for a trip by having user enter expenditures by category.

2.2 OPERATING ENVIRONMENT

Since the software is to be developed for maximum portability and ease of use What’s There?” is to be developed as a Web app, requiring either a computer or mobile device of any type that can access the internet with any modern browser.

Modern browser defined as:

Chrome 60 or later

Firefox 55 or later

Safari 10.1 or later

Opera 47 or later

Edge 16 or later

iOS Safari 10.3 or later

Android Browser 56 or later

Chrome for Android 61 or later

2.3 DESIGN AND IMPLEMENTATION CONSTRAINTS

Since this app is a class project and not intended for a full-scale development team, there are a few constraints that differentiate it from one of larger scale. Since the cities will require hardcoded information in places where a feed isn't/can't be used, specific hardcoded data will be required. The constraint this adds is that even if we are able to generate city pages for any given city, we will not have time to add full support for more than 3-4 cities. A working version of the program should be complete in about a month's time.

2.4 USER DOCUMENTATION

A user manual may be useful, but as a website the usage of the app will be straightforward. Changes to the app should be well documented.

2.5 ASSUMPTIONS AND DEPENDENCIES

We assume that we will be able to access an RSS feed or retrieve API data for the information required, and that our first app release will not have full support for a great multitude of cities.

3. FUNCTIONAL REQUIREMENTS

3.1 CITY SEARCH

a. Description and priority

This will allow the user to find the city they want. It will be a search bar that has a text autocomplete feature based on supported cities. This feature is of high priority.

b. Stimulus/response sequence

The user begins to input city name and the autocomplete feature will offer a city suggestion based on supported cities. When they submit their search, it should direct the user to the page for the city selected.

c. Functional requirements

1. The search bar

Must allow the user to add text and submit a search.

2. The search itself

Once the search is submitted, the user must be directed to the page for the specified city. If the user searches with invalid input, an error message should be displayed on the search page.

3. Autocorrect

A feature to be added if time allows. If the user begins to type “Dallas”, this feature will display the rest of the text of “Dallas, TX” as a search option. Only supported cities will display like this.

3.2 ACCOUNT CREATION AND LOG IN/LOG OUT

a. Description and priority

This will enable features such as home city and favorites. Account is not required to use the app, because of this, this feature is of medium priority. Although it is not required for the app to be used, it is a preferred feature.

b. Stimulus/response sequence

User should have the option to create an account, and their login information must be saved to a database. When logging in, the username and password should be checked with the database of user info, and log them in if correct or provide an error message if not. Log out should simply return the user to an instance of the site where the account features are not available and the log in option is available.

c. Functional Requirements

1. Connect to database

Must be able to add data to and query a database, and compare user data with the data stored in it.

2. Return to logged out version of site

When the user logs off, buttons for account features will not be available. Log in option should return.

3. Log in

Log the user into a version of the site that displays their name, and has the account feature buttons on the navigation bar available. Uses the database(s) to grab their home city and favorites lists.

3.3 WEATHER

a. Description and priority

This feature is high priority. Weather data such as temperature and weather condition, 5-day forecast, and weather alerts may be displayed on the homepage of a city. A dropdown/expand option may be added to display more specific data.

b. Stimulus/response sequence

This data should be grabbed with API calls or RSS feeds when a city page is loaded after search.

c. Functional requirements

1. API call or access RSS feeds

Must be able to access current data from the internet.

2. Dropdown/expand

Section of weather that can be expanded to view more specific current weather/climate data.

3.4 TRAFFIC

a. Description and priority

This feature will access Google's API data to display a map of the city with overlaid traffic data. Medium+ priority.

b. Stimulus/response sequence

User selects traffic data for a city and a map is displayed centered on that city with traffic

Overlaid.

c. Functional requirements

1. API call

Must be able to make an API call and successfully grab the map data.

3.5 PUBLIC TRANSPORT

a. Description and priority

Most of the public transport functionality is still to be determined. As a bare minimum this feature should provide hardcoded links to websites where users can view schedules/reserve tickets. Ideally this feature would grab some sort of current transport data. Medium priority.

b. Stimulus/response sequence

User may select public transport from city page and receive data such as bus or train schedules. If live data isn't grabbed, then the user should be given links to sites where they may make reservations or see schedules.

c. Functional requirements

1. API call or RSS feed

If current data is used, data will have to be retrieved via one of these.

3.6 HOME CITY

a. Description and priority

User may select a city from those available as a home city, that they can directly access from the navigation bar. High priority.

b. Stimulus/response sequence

User selects to save a city as home city, and the site accesses the database(s) to save it. If they choose to load home city, it will redirect them to the page for that city.

c. Functional requirements

1. Connect to database

Must be able to add data to and grab data from the database(s). Some value will represent which city is which that will be used to redirect the page.

2. Redirect to correct city page

Use the value that indicates the specific city to take the user to the correct city homepage.

3.7 FAVORITES

a. Description and priority

Allows the user to save a list of favorite cities. Low priority.

b. Stimulus/response sequence

Works similar to home city, but instead of a button that takes the user directly to the city page, it takes the user to a page that displays the favorites list. From this they select the city they want to view.

c. Functional requirements

1. Connect to database

Must be able to add data to and grab data from the database(s). Some value will represent which city is which that will be used to form a list of the favorite cities.

2. Redirect to correct city page when chosen

Use the value that indicates the specific city to take the user to the correct city homepage.

3. Display favorites list

Must have a way to display selectable cities that are saved in that user's favorites.

3.8 TRIP CALCULATOR

a. Description and priority

The user should be able to enter prices into a table with a list of amenities, and the trip calculator will use this data to calculate an estimated trip price. Relatively simplistic. Low priority.

b. Stimulus/response sequence

User enters prices into the table for each amenity listed, and the calculator responds with a calculated estimate.

c. Functional requirements

1. Calculate the estimate

This will likely be done by letting the user enter estimated amenity prices for a day then multiply by number of days, but is mostly TBD. Simple math.

2. Table

The trip calculator page should display a table where the user enters the data.

3.9 HOTELS AND RESTAURANTS

a. Description and priority

Displays list of local hotels with their prices and ratings or a list of local restaurants with their ratings. Medium- priority.

b. Stimulus/response sequence

User selects one of these options and an API call or RSS feed is made to grab the data to be displayed as a vertical list.

c. Functional requirements

1. Display

Must be able to display the list of hotels or restaurants in a readable fashion.

2. API call or RSS feed

One of these must be used to grab the data needed for the lists.

4. NONFUNCTIONAL REQUIREMENTS

4.1 PERFORMANCE REQUIREMENTS

User must be able to navigate site seamlessly and find their desired content in a matter of seconds. Page and content loading times must be under 1 second for a standard internet connection. User should be able to log in within 3 seconds. Design should be aesthetically pleasing and interactive. UI should be fluid.

4.2 SAFETY REQUIREMENTS

Will use most recent anti-virus system and take all safety precautions to ensure safety of user, client, and server.

4.3 SECURITY REQUIREMENTS

Will follow standard security practices for Web Applications.

4.4 STORAGE REQUIREMENTS

System must be able to handle storing information for up to 100 users. In our case only a handful of users may sign up, but ideally thousands of users could be accommodated. Backup system for user data must be in place.

5. EXTERNAL INTERFACE REQUIREMENTS

5.1 USER INTERFACES

Interface will be centered on the sidebar, as it allows user to navigate through the core features of the application. The display will adapt to the user device. City Page format will be similarly styled and contain a user interface to view and expand weather for a location, and navigate a photo slideshow. A city page's tabs expand to reveal more information about the location, such as a traffic map, list of hotels, list of restaurants, public transportation, and natural geographical information. The design of the tabs will be similar to the navbar, but unique to the city page. Besides the standard touch navigation for the main app, user must also interact with a log in screen and a trip calculator.

5.2 HARDWARE INTERFACES

"What's There?" is a web application and uses a web server and user client hardware combination to process requests for data and present them in an aesthetically pleasing fashion. The app is designed to be supported across a wide variety internet capable devices by using a flexible CSS Grid UI that scales to the device size and purpose. The client device uses a browser to process the layout code and display the application. CSS Grid support is our definition of a "modern browser", a list of which can be found in section 1.

5.3 SOFTWARE INTERFACES

“What’s There?” is developed using HTML5, CSS Grid, Javascript, and utilizes JSON and AJAX for real time data requests. Javascript objects will be distributed to the page and display information. User input to the system determines the data items to be retrieved. PHP will also be used to connect the user database to the webserver.

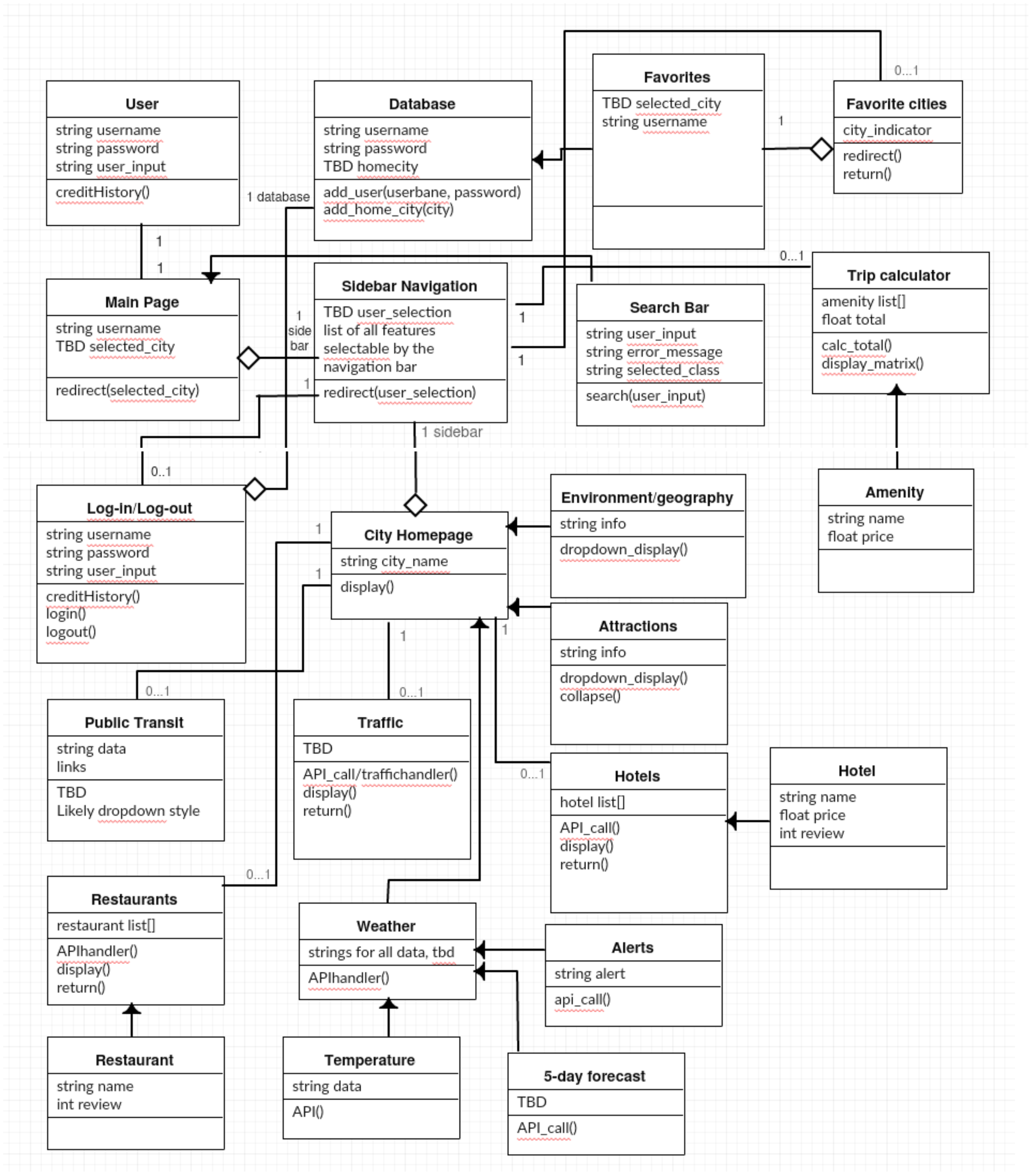
5.4 COMMUNICATION INTERFACES

“What’s There?" is an application consisting of a user client, a HTTP web server to host the application, and a network server that is web-based and created using the PHP (Hypertext Processing) language. The network server exists to retrieve information from the database system that stores user information and user site preferences. The HTTP server will process AJAX API calls from external web services. The network server database should communicate via an encrypted remote connection using SSL to protect customer information.

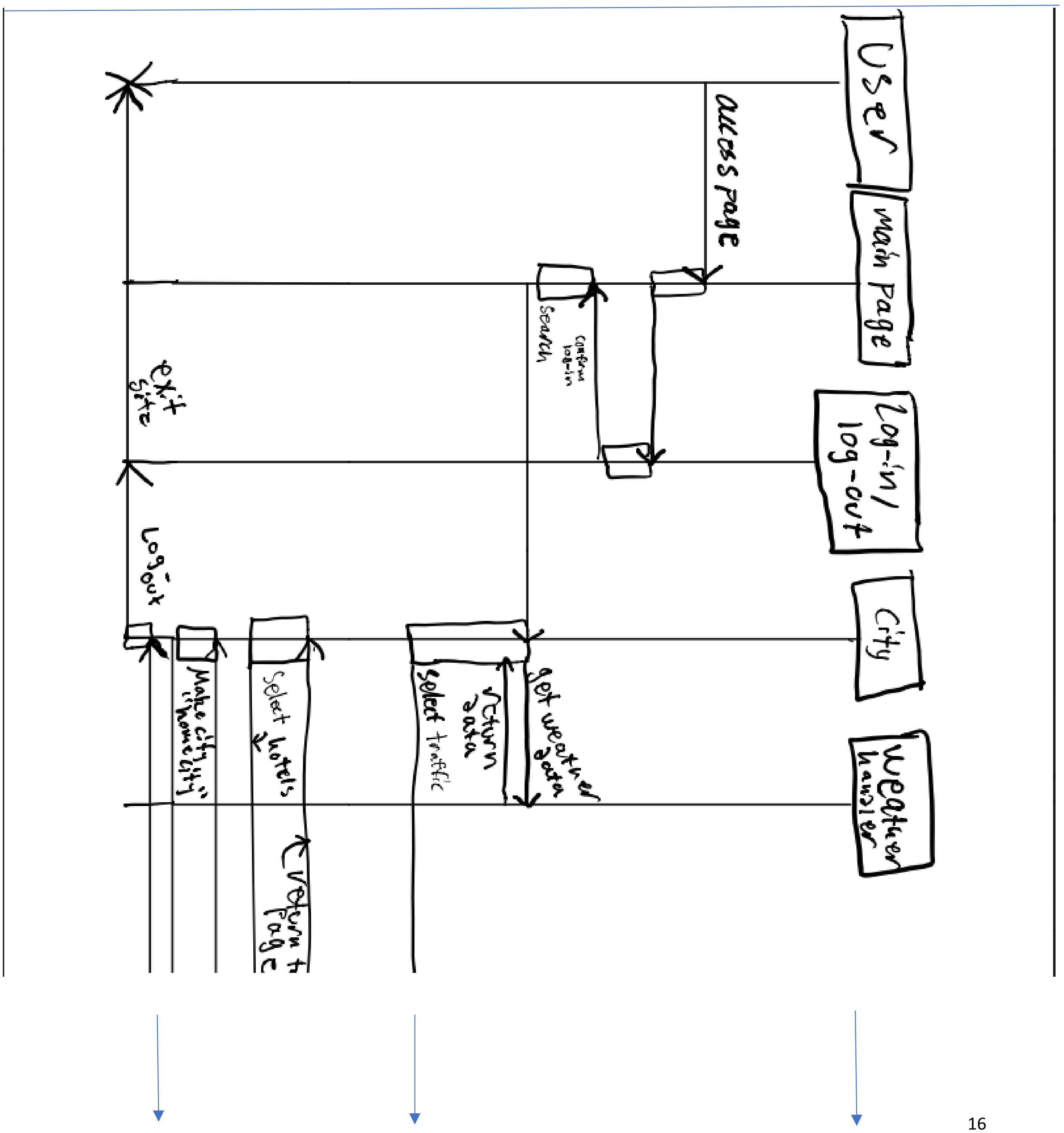
PART II: DELIVERABLE II

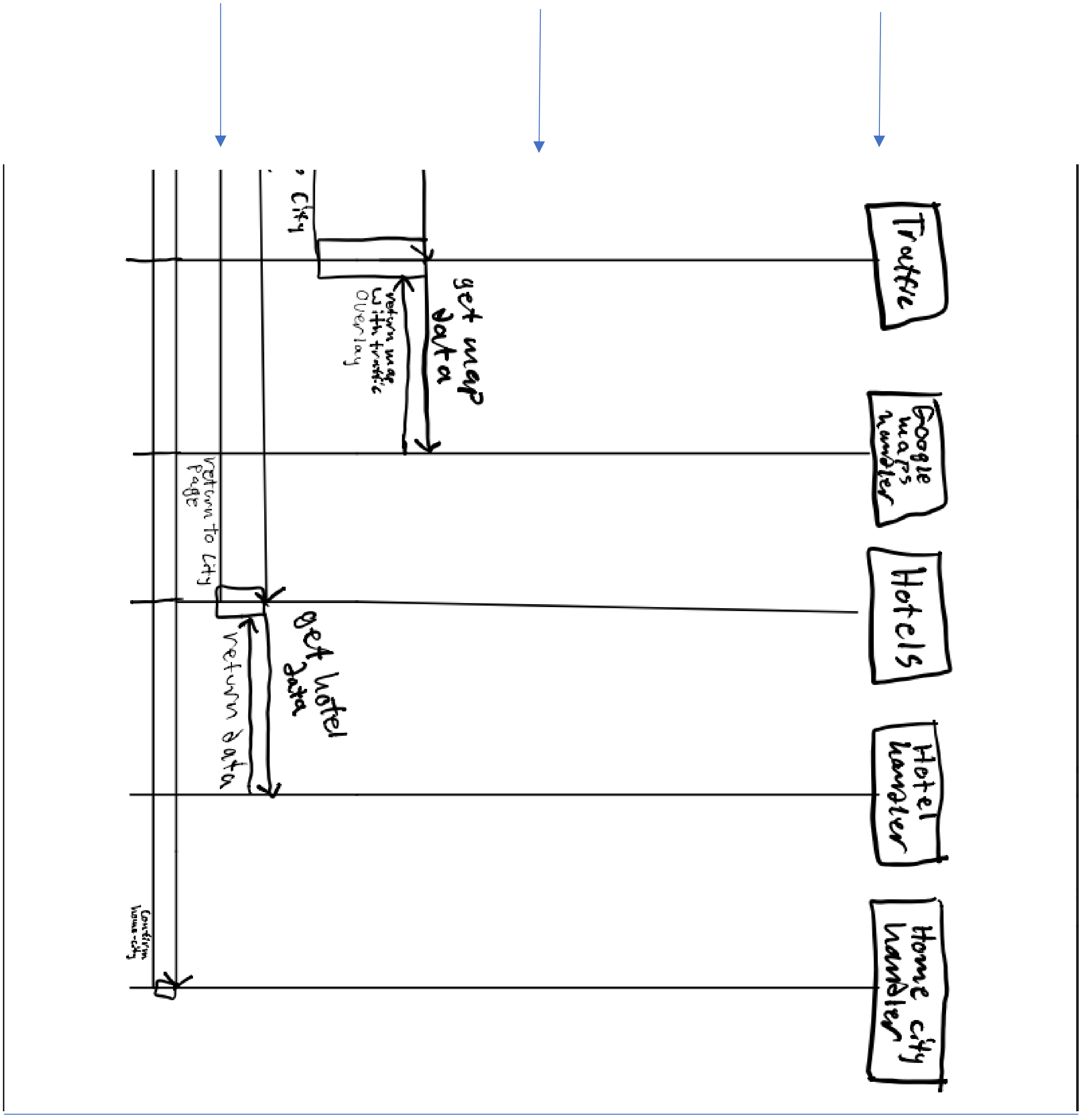
PART 2: UML DIAGRAMS

P2.1 CLASS DIAGRAM

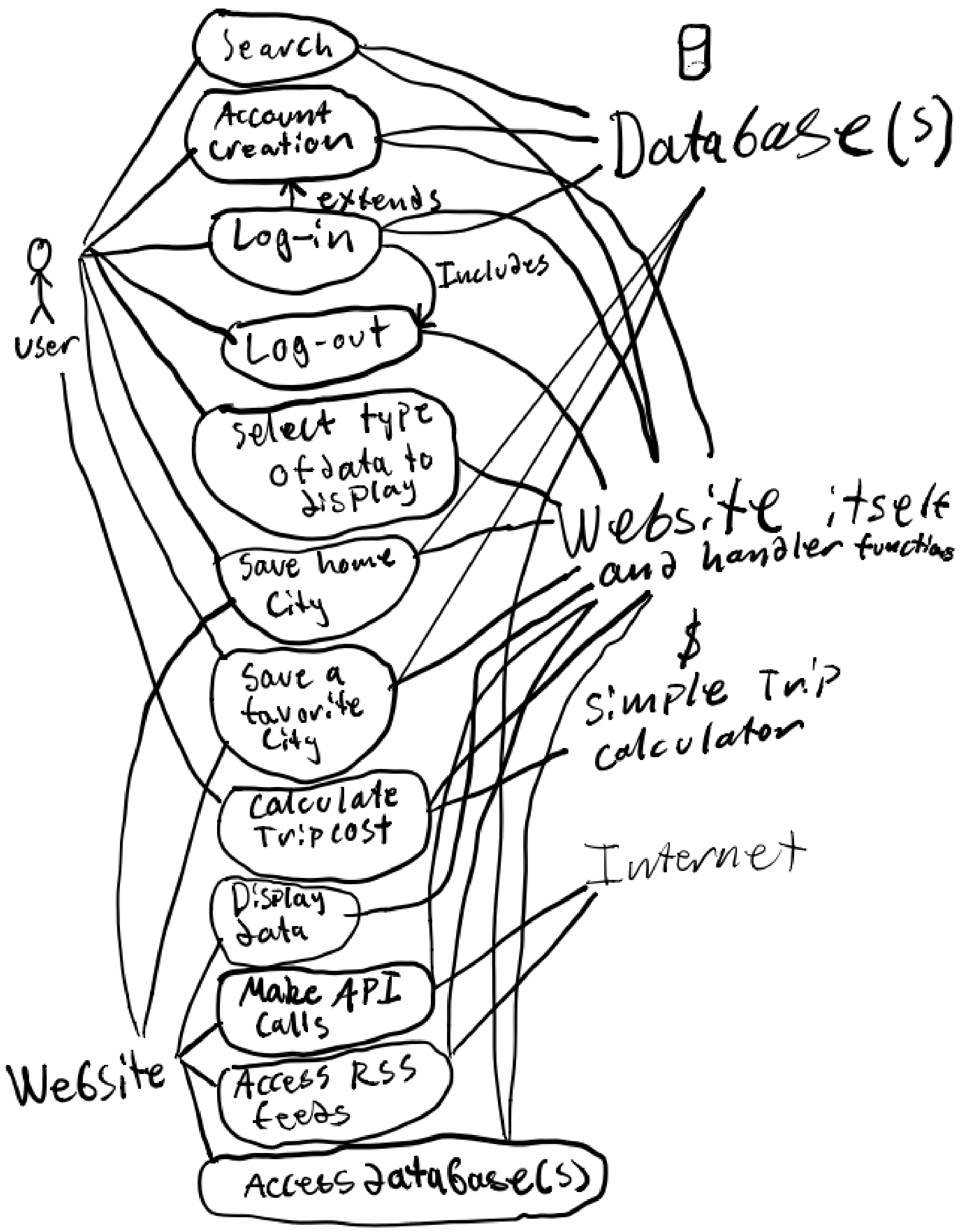


P2.2 SEQUENCE DIAGRAM




Traffic
Google maps handler
Hotels
Hotel handler
Home city handler
get map data
return map with traffic overlay
return to City page
get hotel data
return data
Continue home-city
City

P2.3 USE CASE DIAGRAM



P2.4 USE CASE TEXT

1a. Use Case-01

  a. Signup
  b. Actors: User and Network server
  c. Pre-conditions: User must be logged out

a. User requests to sign up
b. Network server prompts user to enter new account credentials
c. Network server checks credentials
d. Confirm account creation if valid

1b. Error Case-01

1) Signup Error
2) Actors: User and Network server
3) Condition: User enters invalid string for any signup field

a) User sends form information
b) System identifies invalid input and returns error message “Invalid input”
c) System specifies expected input
d) System prompts user to fill out form again

2a. Use Case -02

  a. Login
  b. Actors: User and Network server
  c. Pre-condition: To log in, user must not currently be logged in

a. User requests login with credentials
b. Network Server checks credentials
c. Confirm login if valid

2b. Error Case-02

  a. Login Error
  b. Actors: User and Network server
  c. Condition: User enters invalid string for username or password login fields

a. User sends form information
b. System identifies invalid input and returns error message “Incorrect username or password”
c. System prompts user to fill out form again

3. Use Case-03
    a. <u>Access City Page</u>
    b. Actors: User and Web server
    c. Pre-condition: To access a city page, user must select city from list, or by direct search.

  a. User requests to load a city page
  b. Web server gathers weather AJAX data
  c. Web server responds to request and loads city page

4. Use Case-04
    a. <u>Access Traffic map</u>
    b. Actors: User and Google Maps API
    c. Pre-condition: User must be on city page to access its traffic map

  a. User request page to load traffic map
  b. Google Maps API receives request and processes
  c. Google Maps API returns map with traffic overlay

5. Use Case-05
    a. <u>Access Restaurant and Hotel data</u>
    b. Actors: User and Google Places API
    c. Pre-condition: User must be on city page to access restaurants and hotels

  a. User request page to load list of Hotels or restaurants
  b. Google Places API receives request and processes
  c. Google Places API returns place information,
  d. Client device processes data and formats element to display in list format to user.

6. Use Case-06
    a. <u>Expand and minimize embedded data for geographical and natural information</u>
    b. Actors: User and User Device Client
    c. Pre-condition: User must be on city page to access information

  a. User request page to display minimized page information through a tab
  b. Client processes CSS code to expand and change displayed information on the page.
  c. User request page to minimize expanded page information through a tab
  d. Client processes request and updates displayed CSS to hide information tab.

7. Use Case-07
    a. <u>Save user home city</u>
    b. Actors: User and Home City handler
    c. Pre-condition: User must be logged in

a. User requests to save current city page as home city
b. Home city handler sends updated information to database
c. Home city handler updates home page and confirms change

8. Use Case-08
    a. <u>Logout</u>
    b. Actors: User and Network server
    c. Pre-condition: User must be logged in

a. User requests to logout
b. Network server saves current profile data and logs user out.

PART 3: TEST PLAN

P3.1 UI AND DEVICE COMPATIBILITY

a) Navbar

   i) Aesthetic

      (1) Test icon position on desktop and small media

      (2) Test sidebar layout on desktop and small media

      (3) Test if layout repositions correctly for small media

   ii) Functional

      (1) Test that links are working and pointing to correct page

      (2) Test that responsive navbar elements are working as intended

      (3) Assess if navbar layout is successful in allowing user to get around app.

b) Page content

   i) Home Page

(1) Test Placement of main logo, search bar, and footer across all forms of media.

(2) Test search bar and logo size

ii) City Homepage

(1) Test placement of City name, location and weather information at the top of the page.

(2) Test dropdown for extended weather information

(3) Test buttons to expand and navigate to other city information

iii) Account and Favorites

(1) Test the saving of a home city

(2) Test saving a user specific collection of favorite cities

c) Maps and interactive elements

i) Test that map is correctly positioned on the page on all devices

ii) Test that map is responsive and working as intended

d) Evaluate based on interactivity, layout, readability, aesthetics, display characteristics, time sensitivity, personalization, and accessibility.

P3.2 DATA HANDLING

a) API with JSON

i) OpenWeather

(1) Check a successful connection and return of requested data

(2) Check weather data is being saved

ii) Google Places

(1) Check successful connection and return of requested data

iii) Google Maps

(1) Check successful connection and return of requested data

(2) Check successful display of map

iv) Google Traffic

(1) Check successful connection and return of requested data

(2) Check successful display of map

b) SQL and PHP database information

   i) Check that login data is being saved to database

   ii) Check that account information (home city/favorites/preferences) is saved for each user

   iii) Check for successful login and restoration of saved features

c) Logs

   i) Test that data about the site is being stored in database

P3.3 LOGIN/PROFILE SYSTEM

a) Test sign up system
   i) Check that system only allows certain characters for username and password
   ii) Incorrect Input: User input is not within allowed parameters, return error message
   iii) Correct Input: User signup is confirmed and user is logged in.
b) Test login system
   i) Incorrect: Return error message if incorrect username or password was entered
   ii) Correct: If user input matches user authentication in database, confirm login.
   iii) Test password reset system

P3.4 SECURITY

a) Security concerns and testing methods to be evaluated

P3.5 PERFORMANCE

a) Test that data structures implemented in app are efficient and app does not suffer long wait and/or loading times.
b) Test high load scenarios

PART 4: RISK MANAGEMENT

Updated Risks (top 5)

1. Time management - Time is going by fast, and code inspection is not far away. We need to stick to the schedule to make sure that the app gets finished by inspection.

2. Communication - Group members need to be using slack more to curate information, ideas, and plan.

3. Web Server Issues - Inexperienced at server side programming, which means that fixes and functionality implementation may take longer than we expect. Setting up the webserver and database may be a challenge.

4. Technical problems/difficulties - As all developers know, bugs are going to happen. They can delay a team and deal a massive blow to your project in the long run, robbing you of time to add and perfect features. There is always a chance of encountering problems with other software such as your IDE, repository, and individual files in a large project.

5. Schedule Conflicts - The group members' schedules conflict often, so it is difficult to plan meetings.

Monitoring - We have been mitigating our risk by holding regular meetings, both in and outside of class. Team performance needs to improve to meet deadlines. Will require more work than some members of group are currently contributing.

Deliverable II Re-Evaluation - Many development tasks for the project are not difficult, but require dedication to learning the core concepts behind them. Each member of the group has a personal responsibility to research and understand the requirements for the app. Group has had very productive meetings and planning sessions, leading to a change in requirements and project specifications.

Contingency update - At this point, we have a better understanding of the programming requirements for our desired functionalities. As a result, we have made some changes to our features to accommodate our timeline. Further contingency plans may include omitting the trip calculator or the login system if we are behind schedule, or adding a travel calendar and other features like News and gas prices if we are ahead of schedule and have time to add more features.

PART 5: UPDATED PROJECT PLAN

Only small updates since deliverable I.

9/13: Have a complete list of features and a good idea what the final product will be

9/18: Be familiar with visual studio and RSS feeds

9/20: Have requirement specifications laid out

9/22: Have begun programming, the basics

9/27: UML design

9/29: Create first files of application, main html and CSS

10/2: Have actual functionality for the search, returns a result

10/4: Test Plan laid out

10/6: Finalize design layout

10/11: Have rest of Del. 2 finished

10/12: Begin heavy coding period. Focus on sidebar functionality and sidebar navigation destinations

10/20: Have sidebar functionality and city list completed, begin working on city homepage html and CSS

10/23: Have top half of city homepage finished, begin working on main city page information tabs

10/27: Have a working city list, begin working on city page details, begin working on login system and database

11/3: Have most front-end development done, begin working on user account functionality and trip calculator

11/6: App should be mostly put together, begin preparation for del. 3 by going over previous

code. Continue working on polishing code.

11/12: Have code ready for inspection

11/13: Testing and polishing

11/27: Completion

1PART 6: MEETING MINUTES

10.1 9/21

Elicitation suggestions:

- Links to sites for transportation or restaurants for reviews or reservations.

- Hotel and restaurant rating system.

- City categories, such as residential, tourist, commercial.

- Databases.

Discussion:

- Agree with links and databases.

10.2 9/26

In-class meeting: Decided C# would be preferable over C++, still a desktop app. Decided on meeting on 9/28 at 5:30PM to work on DII. Wrote most of functional requirements this meeting.

10.3 9/28

In-class meeting:

- Discussed switching to web app. Using Atom text editor.

- Went over installing sourcetree and accessing the web app.

- Evening meeting: Did non-functional requirements and interfaces section.

- Did almost all of use case diagram.

10.4 10/3

Discussion:

- Discussed decision to move to a web app. Nasser showed a demonstration

of a C# program that can get weather data for a city. We now understand how to

get live data and what format it will be in.

10.5 10/5

In-class discussion:

- Review of class diagrams. Discussed the visual/user interface design.

- Made preliminary visual design for the entire web app (implementing this has begun).

- Decided that primary navigation will be with a side-bar on the left of the screen.

Decided log-in should be optional, can use site as guest, having an account enables: home city, favorites.

Evening meeting:

Created class diagram, all but the side-bar functionalities included so far.

From this and our design decisions in class, we have a nearly exact idea how the

program will function.

10.6 10/6

Created sequence diagram. Decided it would include Log-in, city select, weather, traffic, news, hotel prices, home city save. Needs polishing, but most of the diagram is complete.

10.7 10/10

Discussed city homepage design. Hardcoded information should be dropdown boxes. Picture slideshow.

PART 7: PROJECT PROGRESS REPORT

Since the presentation, we have decided to build a web app instead of a C++ desktop app. We will make use of HTML+CSS, Javascript, and PHP, and will use Atom text editor. Changes to design include: hotel and restaurant price check, sidebar navigation, optional account creation (with home city and favorites), and plan on map data for traffic. A basic trip calculator will be included. Scratched gas prices from features for now and news data is uncertain. Used a slack day for deliverable 2. The app itself has already been started and we currently have the main/search page mostly set up and the navigation bar started. We have the entire visual layout and navigation design concepts in mind.

PART 8: CONTRIBUTION TABLE

Member name	Contribution description	Overall Contribution (%)	Note (if applicable)
Peter Menchu	Put final document together and did most of the requirements sections, input on functional requirements, did most of UML documents. Wrote project report and minutes.	30%	
Garrett Brumley	Organized repository, input on diagrams, did use cases and test cases, did nonfunctional and interfaces, references, input on diagrams. Updated risks and plan.	30%	
Nasser Alqudaihi	Input on functional requirements, helped build class and sequence diagrams, provided input on how data will actually be retrieved and used/displayed, input on design.	20%	
Joseph Vo	Input on functional requirements, helped build use case, class, and sequence diagrams, input on design.	20%	