Project Proposal for Nightsky Web Application

(CS 275 Section 001)

Brandon Au

Ryan Canavan

Jerry DiBello

February 24, 2017

# 1. Introduction

# 1.1 Purpose

The purpose of this project proposal is to discuss the idea of the Nightsky project to the instructors and teaching assistants of CS 275 winter term 2017. This document is to be referred to by the development team and project manager of Nightsky for what the web application should do by the end of the term in March.

# 1.2 Project Scope

Nightsky will be a web application that shall display information of upcoming space eventss such as lunar eclipses, meteor showers, comets, and which planets will be visible during the nighttime. Nightsky will use an external API called [Predict the Sky](http://predictthesky.org/), which is a web service that NASA provides. Nightsky shall be able communicate with the Predict the Sky application program interface to fill a database of the different types of space events at a certain location. The webpage should then allow the user to choose which event they are interested in and then populate the content of the webpage with the dates and names of the space events.

# 1.3 Definitions, Abbreviations, and Acronyms

API – Application Program Interface

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

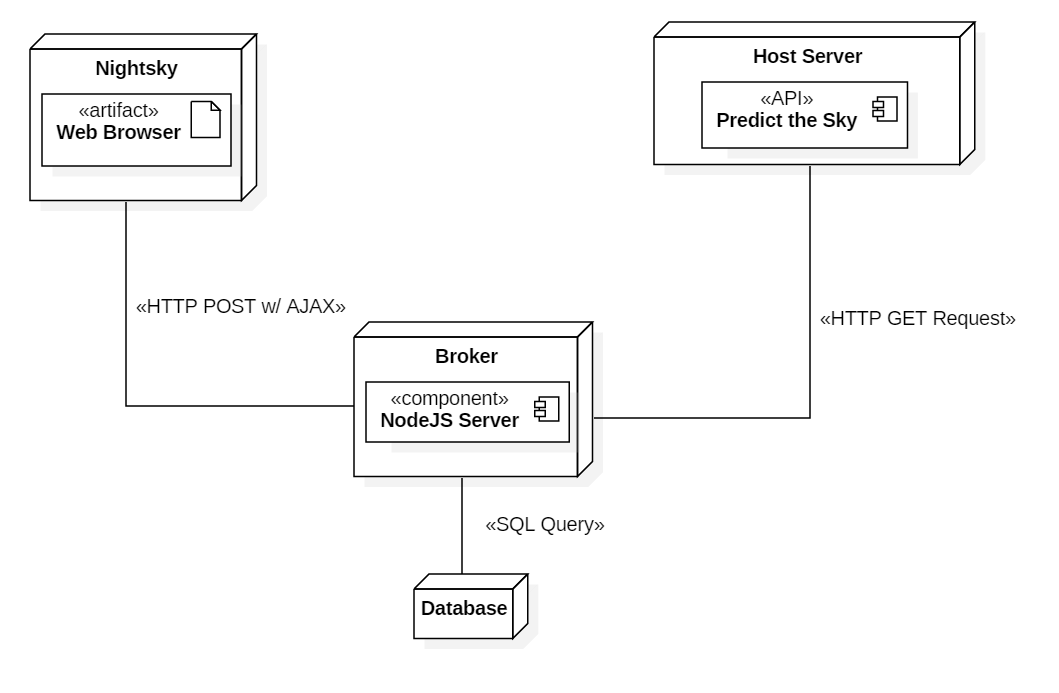
# 1.4 Document Overview

The rest of the document will discuss the architecture of the project in section 2, functional requirements in section 3, and the development schedule of the project in section 4.

# 2. Nightsky System Architecture

# Technologies to be Used:

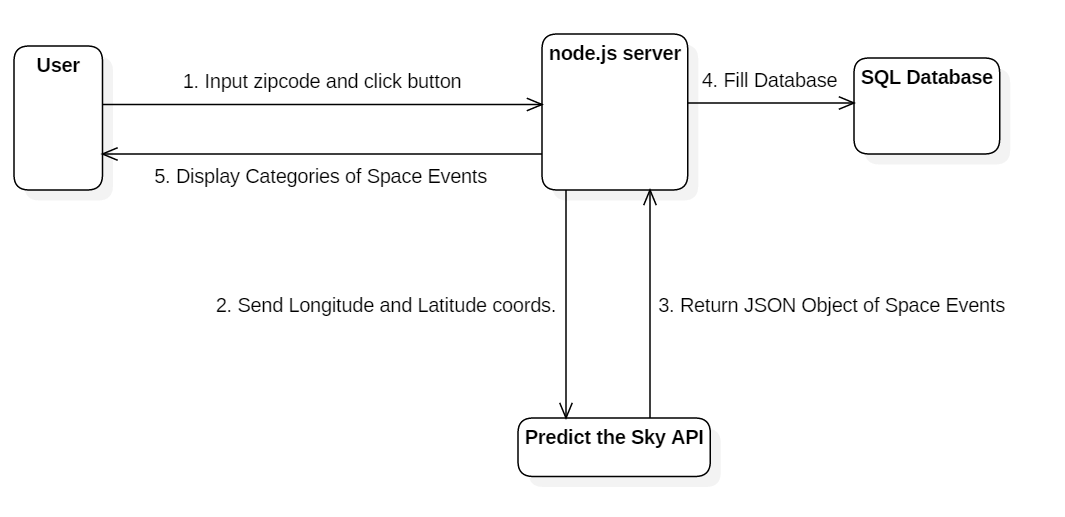
* HTML – Provides structure to the client webpage. The input text field and buttons will be declared in an HTML file. HTML5 provides the geolocation feature which can be used for receiving longitudinal and latitudinal coordinates.
* CSS – Provides style attributes to elements of the HTML file.
* JavaScript – Provides functionality to the webpage which makes it more interactive.
* jQuery – Provides the AJAX object to perform asynchronous calls to the node.js server.
* Node.js – Utility to send client messages that include longitudinal and latitudinal coordinates to the Predict the Sky API.
* MySQL Database – Stores information that Predict the Sky returns.



*Broker Architecture Diagram of Nightsky*

The Nightsky web application will be based on the broker architecture style, as displayed in the diagram above. The broker architecture style is a type of a client-server architecture style. The main component of this style is the broker. The broker is in charge of routing the client’s messages to the correct server.

The client side portion of Nightsky will be the interactive webpage. The inputted zipcode and the click of a button will start an asynchronous call to a specific endpoint of the broker. The broker is a node.js server that makes a HTTP requests to the Predict the Sky API. The broker will handle the response from the API and populate the database with the different space events listed in the JSON response. We decided to use a broker to handle the HTTP requests and responses because it allows for easy extensibility, modifiability, and encapsulates changes to the backend to the user. If the web application were to use another API, a new endpoint can easily be added to the node.js server.



*Data Flow Diagram of Nightsky*

# 3. Software Requirements

# 3.1 Functional Requirements

**Feature 3.1.1 Accepting 5 Digit Zipcode for Location**

REQ-3.1.1.1 Nightsky shall only accept zipcodes that belong to the United

States.

REQ-3.1.1.2 Nightsky shall be able to find latitudinal coordinates based on

zipcode input.

REQ-3.1.1.3 Nightsky shall be able to find longitudinal coordinates based on

zipcode input.

REQ-3.1.1.4 Nightsky shall notify the user if the zipcode is invalid.

**Feature 3.1.2 Communicating with the Predict the Sky API**

REQ-3.1.2.1 Nightsky shall use a HTTP GET request to the Predict the Sky

API.

REQ-3.1.2.2 Nightsky shall be able to parse the JSON object response for

the space object’s ID, name and category attribute.

REQ-3.1.2.3 Nightsky shall be able to parse the JSON object response for the

space object’s start and end timestamp attribute.

REQ-3.1.2.4 Nightsky shall send the longitudinal coordinates of the zipcode-

based location to the API.

REQ-3.1.2.5 Nightsky shall send the latitudinal coordinates of the zipcode-

based location to the API.

REQ-3.1.2.5 Nightsky shall display a message if the API cannot be reached.

**Feature 3.1.3 User Chooses what Space Event to View**

REQ-3.1.3.1 Nightsky shall display a dropdown menu of the upcoming space

events (planets, lunar eclipses, auroras, meteor showers, comets.)

REQ-3.1.3.2 When the user selects an option from the dropdown menu, the

Nightsky system shall display the specific details of the event(s). For example, if the planets option is selected, then the webpage could show Jupiter, Saturn, Venus as the space events that are upcoming.

REQ-3.1.3.3 Nightsky shall change the information depending on the selection

of the dropdown menu.

REQ-3.1.3.4 Nightsky shall display a message notifying the user if there are no

upcoming space events.

**Feature 3.1.4 Acquiring User Location via Geolocation**

REQ-3.1.4.1 Nightsky shall be able to find latitudinal coordinates using

geolocation.

REQ-3.1.4.2 Nightsky shall be able to find longitudinal coordinates using

geolocation.

REQ-3.1.4.3 Nightsky shall inform the user if the system does not have

permissions to use their geolocation.

# 4. Project Schedule

# 4.1 Gantt Chart

See next page.

# 4.2 Team Member Roles

**Note:** These roles are bound to change and overlap each other as the development process continues.

Brandon Au – Maintains documentation and ensures that deliverables are handed it on time. Also

in charge of the user interface and the structure of the web application.

Ryan Canavan – Developer of the communications between the node.js server and the database.

In charge of filling in the database according to the JSON response from the API and categorizing the information into different tables.

Jerry DiBello – Developer of the communications between node.js server and the Predict the Sky

API. In charge of sending the longitude and latitude coordinates of the user’s

location to the API.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tasks** | **Member Responsible** | **Start Date (MM/DD/YY)** | **Projected End Date (MM/DD/YY)** | **Progress (%)** |
| **Setup User Interface of Nightsky Webpage** |  |  |  |  |
| HTML File | Brandon | 2/27/2017 | 3/5/2017 | 0 |
| CSS File | Brandon | 2/27/2017 | 3/5/2017 | 0 |
|  |  |  |  |  |
| **Develop node.js server** |  |  |  |  |
| Get node.js server to communicate with client | Brandon | 3/6/2017 | 3/12/2017 | 0 |
| Get node.js server to communicate with Predict the Sky API | Jerry | 3/13/2017 | 3/19/2017 | 0 |
| Get node.js server to communicate with MySQL Database | Ryan | 3/13/2017 | 3/19/2017 |  |
|  |  |  |  |  |
| **Develop Client-Side JavaScript** |  |  |  |  |
| Find longitude and latitude coordinates based on given input | Jerry | 3/6/2017 | 3/12/2017 | 0 |
| Find longitude and latitude coordinates based on geolocation | Jerry | 3/6/2017 | 3/12/2017 | 0 |
| AJAX Call to node.js server sending long. And lat. Coordinates | Jerry | 3/13/2017 | 3/19/2017 | 0 |
|  |  |  |  |  |
| **Database Management** |  |  |  |  |
| Fill the database according to JSON Response from API | Ryan | 3/13/2017 | 3/19/2017 | 0 |
| Pull categories from database and create dropdown on webpage | Ryan | 3/13/2017 | 3/19/2017 | 0 |
| Load items from category onto the webpage | Jerry | 3/13/2017 | 3/19/2017 | 0 |
|  |  |  |  |  |
| **Documentation** |  |  |  |  |
| Interim Status Report | Brandon | 3/6/2017 | 3/9/2017 | 0 |
| Final Presentation | Brandon | 3/13/2017 | 3/20/2017 | 0 |