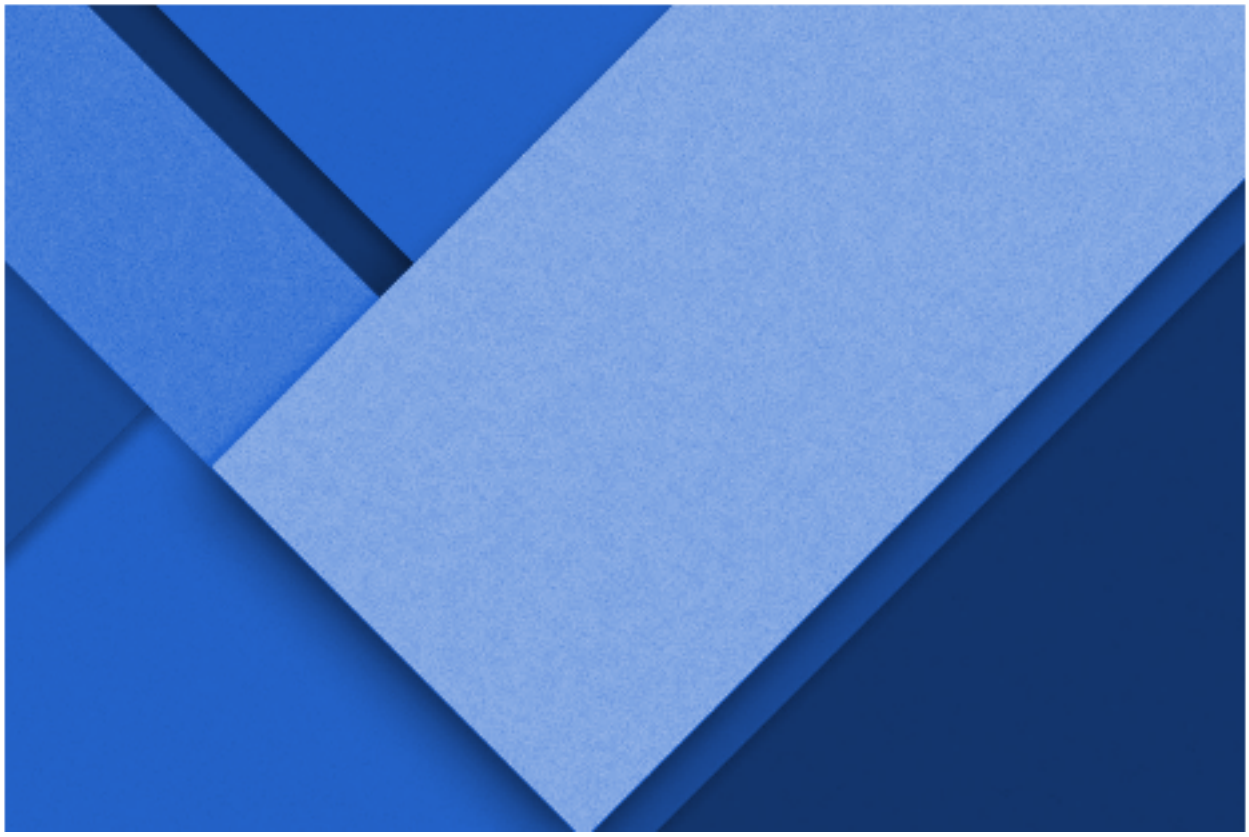


Deliverable 2

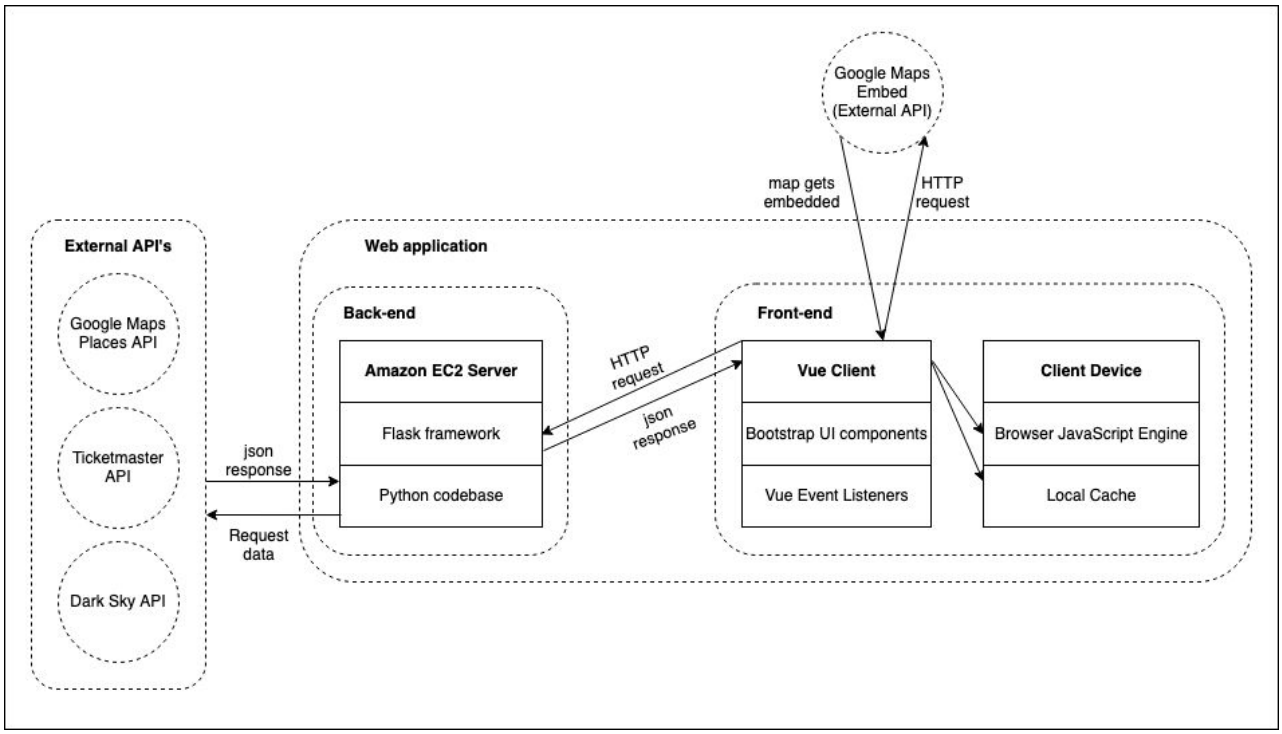
27.03.2020

Houston Rockets

SENG2021

Luke Chen, Kenneth Mejico, Vicknesh Ravikumar, Mubarrat Hossain

Software Architecture



External Data Sources

The application will be accessing various external data sources in order to meet the user's requirements. The system will be accessing the Ticketmaster API to get event information including dates, location, prices, artists etc. The Dark Sky API will be used to get weather information up to a 7 day forecast, and the Google Maps APIs (Embed and Places) will be accessed to obtain travel directions by car, foot, or public transport as well as venue location information such as user ratings.

Language Choices

When deciding which language to use for each component of the software architecture, a variety of options were considered and evaluated to choose the languages which were most suitable for the needs of the application *eventmaster*.

Back-end Development

For back-end development, Python, Java/C#, and C/C++ were evaluated as these languages are currently considered some of the most widely used languages for web app development.

Python

Advantages	Disadvantages

• Extensive support libraries and frameworks which reduces the amount of code needed to be written and simplifies the process - makes development more efficient and increases productivity
  ○ Web development frameworks include Django, Flask, CherryPy, Bottle etc.
  ○ Offers scalability - can quickly add features to prototypes
• Readability - easy to learn, understand, and code; indentation is mandatory. This makes it good for collaboration
• Supports both procedural and object-oriented programming - reusable code which reduces time to market making it good for developing a prototype in a limited time frame
• Portable - can be run on multiple platforms without making any changes to the code
• Large community of users
• Automatic memory management
	• Speed - slow compared to languages like Java or C/C++. Needs an interpreter to execute code line by line. High-level language so it is not closer to hardware compared to C/C++.
	• Weak in mobile development
	• High memory consumption due to the flexibility of data types
	• Limitations with Python's database access.
	• Has errors that only show up in runtime as the language is dynamically typed

Java/C#

Advantages	Disadvantages
• Statically typed so errors can be weeded out during compilation rather than waiting for runtime and testing all the different paths the program can take • Higher performance speed than python • Large number of libraries and frameworks available • Large community of users • Object oriented - offers maintainability and modularity benefits	• Vulnerable to security threats • Syntax isn't as simple - not as readable as python making it harder for collaboration • Strongly typed - When you declare an object or a variable, you must say what it is first. This can be quite cumbersome and the maintenance benefits are not significant for a project of our scale • Code must be compiled after every change before running - adds time to development

C/C++

Advantages	Disadvantages
• Low level programming language - very efficient, high performance • Statically typed so errors can be weeded out during compilation rather than waiting for runtime and testing all the different paths the program can take • Large number of libraries and frameworks available	• May be too complex in large high-level programs • Commonly used for platform specific applications and not very popular for web development • Need to manually manage memory

Summary

Feature	Python	Java/C#	C/C++	Relevance
Web frameworks	Flask, Django - easier to use	Can be more complex and harder to use than something like Flask	Not much support, more suitable for mid-level/low-level applications	**Very High**

Ease of use	Very Good - good readability, automatic typing, automatic memory management, flexible structure	Good - automatic memory management but strongly typed language and object-oriented structure	Moderate - need to manually manage memory, strong typed	**High**
Performance	OK performance - slow due to interpreter and it being dynamically typed	Good performance - faster than python but slower than C/C++	Very efficient, very good performance	**Low - Moderate**

After evaluating all three languages, the team decided to use Python for the application's backend development due to its readability, modularity, portability and extensive library support making it ideal for developing a simple web application in a short time-frame. Python's modularity and extensive library support would reduce the amount of code required, shortening the development process whilst making it faster and easier to build and update prototypes. Collaboration would also be made easier through the readability of the language, allowing team members to easily understand other members' code.

Whilst Python does cause lower performance speeds due to the need of an interpreter to run the file line-by-line, the speed difference compared to Java/C# will be minimal due to the relatively small scale of the application and backend programming involved. Additionally, Python's poor support for database access does not affect the current application as all event data will be accessed and stored on external sources with the exception of user data. Furthermore, Python's lack of support for mobile development also does not factor into our decision due to *eventmaster* being developed as a web application.

Front-end Development

The languages of JavaScript, HTML, and CSS were chosen for frontend development. This was a natural choice due to the nature of the program being a Web Application. There are alternatives to JavaScript, however, most of these alternatives are not widely used and hence do not have the support or features of JavaScript.

Framework Choices

Python Web Framework

Feature	Flask	Django	Relevance to Project
Mapping URLs to views	Yes with support for function and class-based views	Yes with support for function and class-based views	High
Ease of use	Good - easier to learn and set up	Moderate	High
Flexibility	Good - flexible app structure; large availability of extensions	Moderate- forces consistent app structure	Moderate
Performance	Good - smaller and has fewer layers than Django	Almost as good	Moderate
Authentication and Authorisation with account management	No, but can be done through extensions	Yes	Moderate
Input handling, client and server side validation of forms	No, but can be done through extensions	Yes with handling of various security concerns such as CSRF, XSS, and SQL injection	Moderate

After evaluating both Flask and Django web frameworks for Python, Flask was chosen as the preferred framework for the application due to its ease of use and flexibility. Due to the relatively small scale of the application, the many features that Django provides did not seem really relevant to the project. Additionally, the features that are relevant such as input handling and user account management can be implemented in Flask through the use of extensions. Hence, by using Flask, the team is able to easily set up a web server with only the required features for the application, improving both readability and performance.

JavaScript Framework

Feature	Vanilla JavaScript	Vue	React	Relevance
Ease of use	Poor - have to manually write in features such as state management etc.	Good - lightweight, easy to use	Moderate - steep learning curve	High
Maintainability	Poor - have to manually keep UI in sync with state - a lot of code, low readability	Good with separate JavaScript, HTML, CSS files	Good - use of react components	High
State Management	No	Yes	Yes	High
Routing	No	Yes	Yes	High
Virtual DOMs	No	Yes	Yes	Moderate
Server-side rendering	No	Yes	Yes	Moderate
Documentation	Moderate	Good - considered better than React's	Good with large community	Moderate
Scalability	Poor - hard to scale up	Good - both up and down	Good up, not so good down	Low - Moderate

After evaluating the features of Vanilla Javascript, Vue framework, and React framework, the team decided on using the Vue framework for the front end development of the application. This is because Vue (and React) provide state management and routing among other features and components that significantly reduce the lines of code required, assisting the team in releasing a prototype in the given time frame. Additionally, frameworks make it easier to keep the UI in sync with the state of the application through a virtual DOM which improves efficiency especially in cases where data is constantly changing such as when the user applies filters to search results.

Vue was chosen over React as the preferred framework due to its ease of use in setting up as well as its more common structure of separating JavaScript, HTML, and CSS. React also has a steeper learning curve and lacks the ability to scale down making Vue the more suitable framework for a project on a relatively small scale.

Deployment Platform

Feature	AWS	Heroku	GitHub Pages	Relevance
Python Flask Support	Yes	Yes	No	Required
Ease of use	Poor - hard to use, requires manual configuration of server and app	Good - performs features automatically, poor configurability	Good - automatic deployment of Github repository	High
Cost	12 month free trial	Free for small projects	Free with Github account	High
Performance	Very Good	Slow first request	Only supports static content	Moderate
GitHub Integration	Yes	Yes	Yes	Moderate
Deployment Speed	Can be quite a hard process so may take some time	Fast to deploy, easy to make changes and updates after	Fast and easy with Github repository	Moderate

After evaluating the deployment platforms of Amazon Web Services (AWS), Heroku, and GitHub pages, Heroku was chosen as the preferred deployment platform due to its ease of use and rapid deployment, allowing for easy deployment of prototypes and updates to the application. Whilst AWS has high configurability and performance, these advantages are relatively irrelevant to a project of our scale. GitHub pages was ruled out immediately due to its lack of support for Python, our chosen backend language.

Summary of Choices

The chosen software architecture involves using the Python language and Flask framework for the backend whilst utilising the Vue JavaScript framework along with HTML and CSS to develop the frontend. The external sources of Ticketmaster, Dark Sky, and Google Maps APIs will be used to access the data required for the application.

The Python language was chosen due to its readability, modularity and extensive library support which all improve collaboration whilst streamlining the development process through reusable code. Most limitations of the language had minimal impact due to the small scale of the project as well as the sourcing of data externally, mitigating the need for database support. The Flask web framework was chosen for similar reasons being its ease of use, flexibility and lightweight nature outweighing the more complex and structured Django web framework which had many more built-in features that were deemed unnecessary for this application.

Vue was selected as the JavaScript framework for this application due to it being faster to set up and easier to use compared to React. By using a JavaScript framework as opposed to plain JavaScript, the amount of code required is significantly reduced through the many built in features of Vue such as state management and routing. This simplifies the development process whilst also making the application less fragile and more readable, maintainable and extensible.

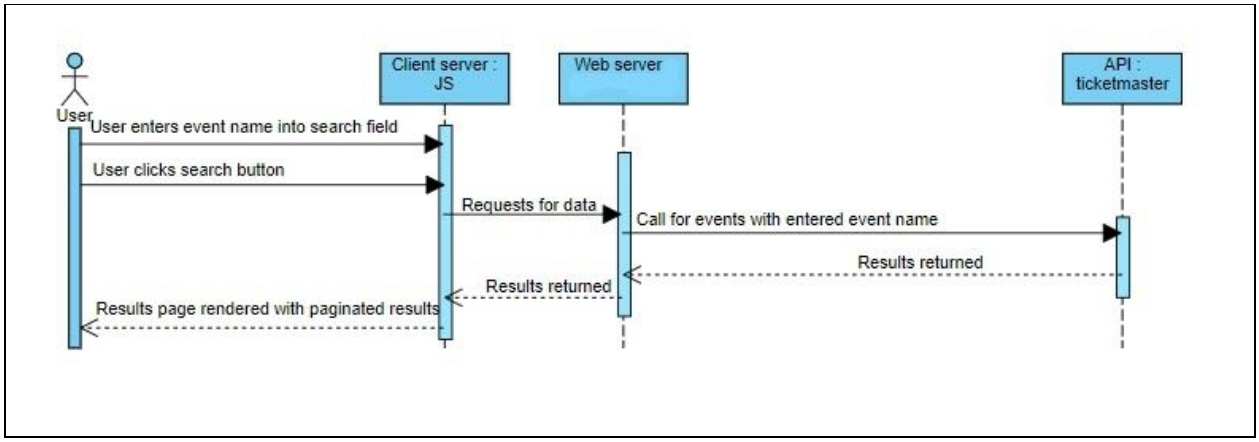
Finally, the team elected to use Heroku as the deployment platform for the application due to its support for Python, as well as the rapid deployment and automatic configurations making it easy to set up and use. This allows the team to focus on the development of the application without needing to manually configure and maintain the server.

As with all the choices made in the software architecture for this project, the team prioritised ease of use and reusability whilst avoiding any unnecessary features to mitigate time spent on steep learning curves and the set up process. These decisions were made to streamline the development process, improve collaboration, and reduce code in order to release a fully functioning application in the given timeframe.

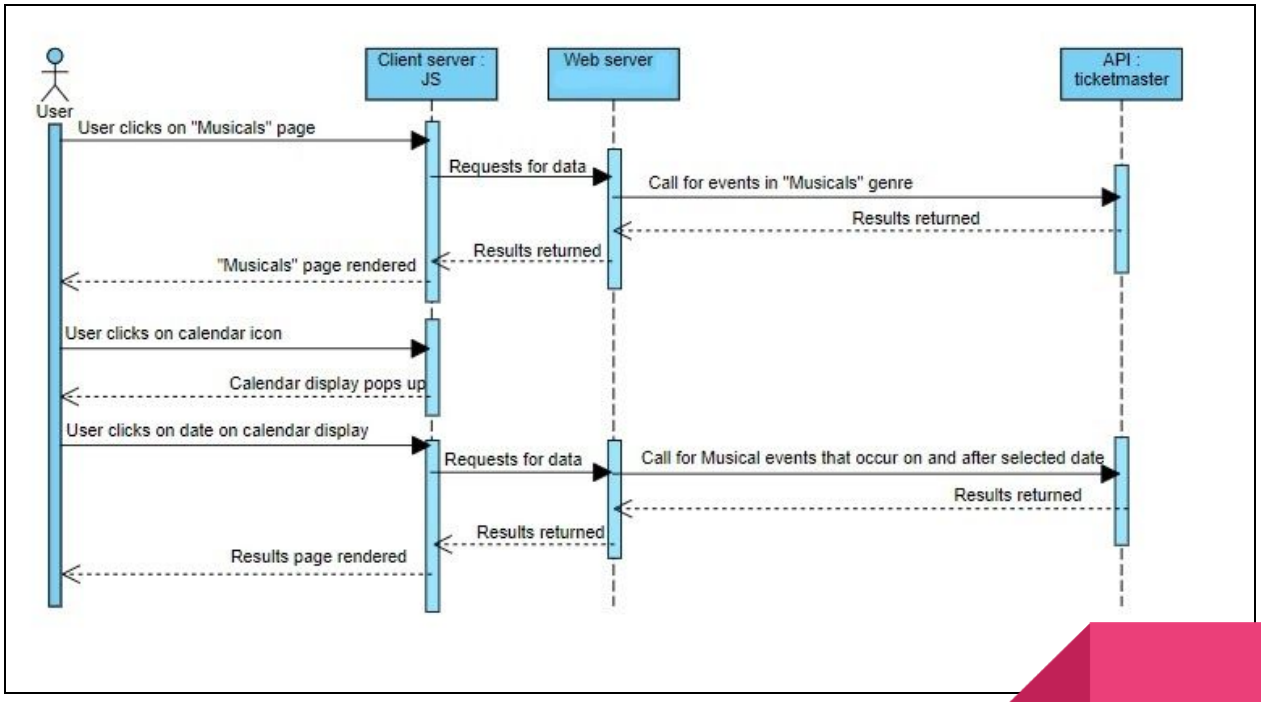
Initial Software Design

UML Sequence Diagrams - User Stories

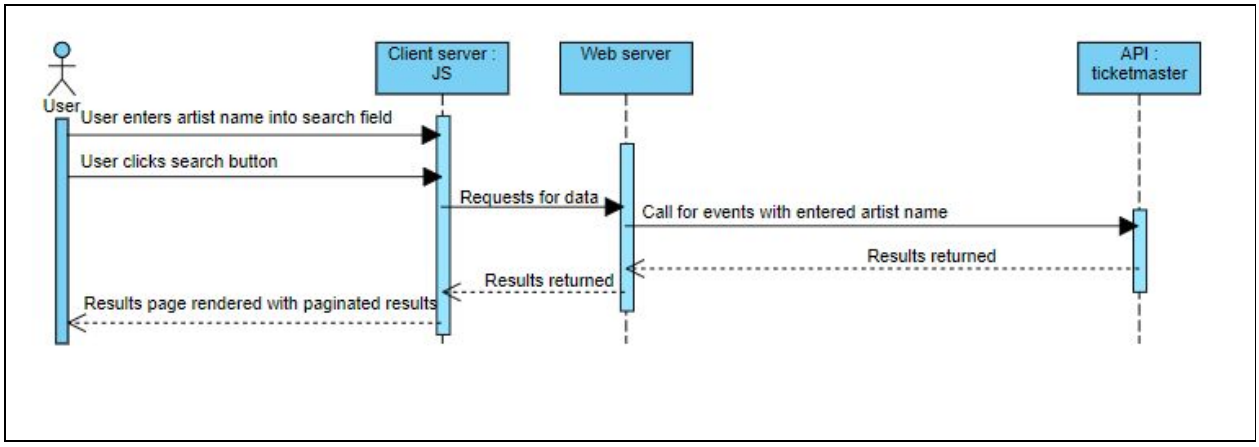
(1A) Feature: Search event by keyword/name



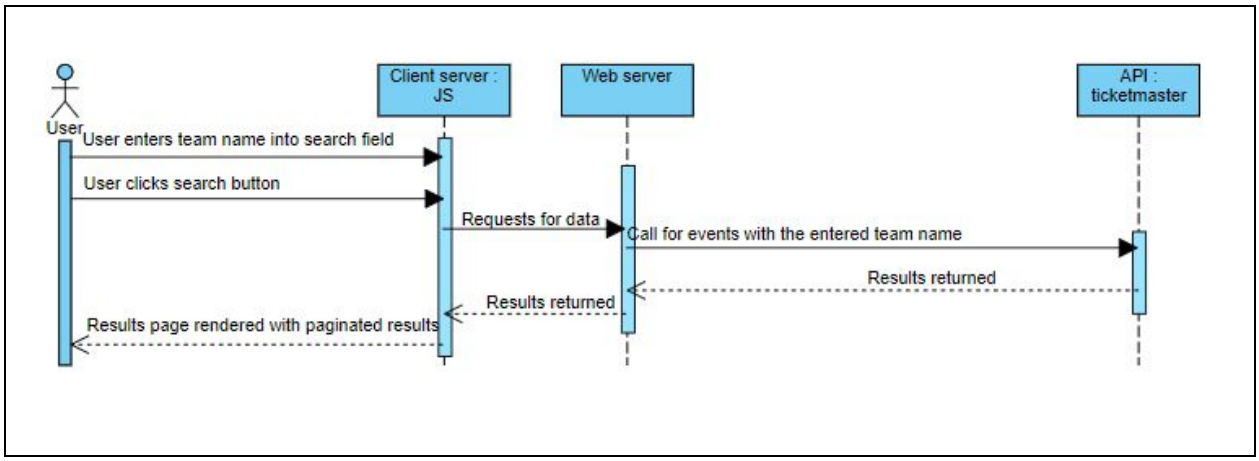
(1B) Feature: Search event by date



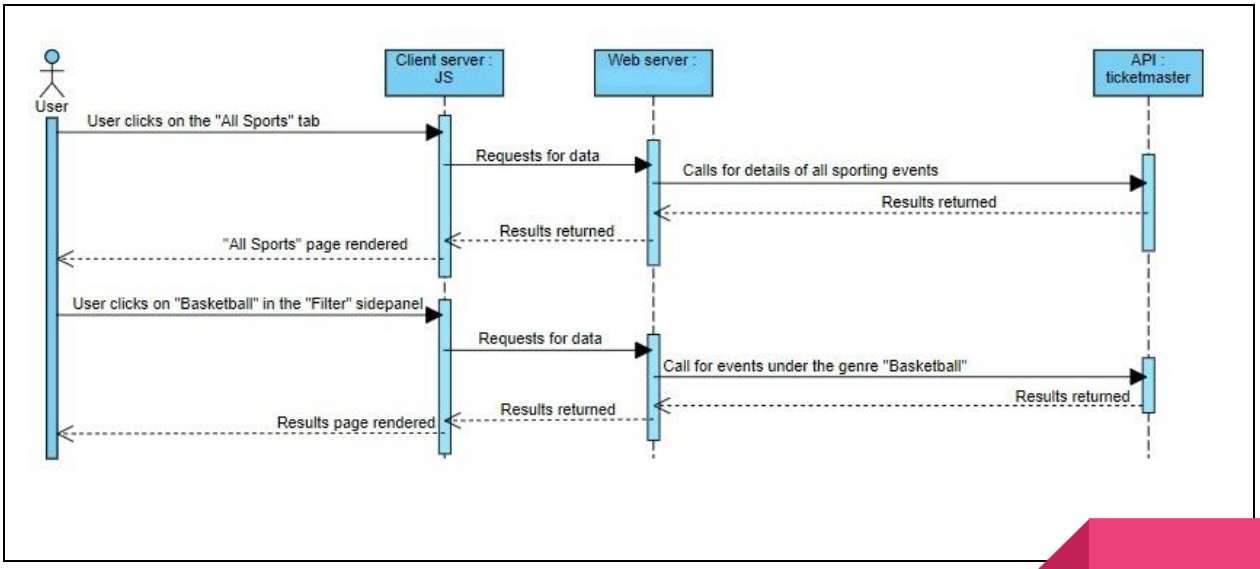
(1C) Feature: Search event by artist



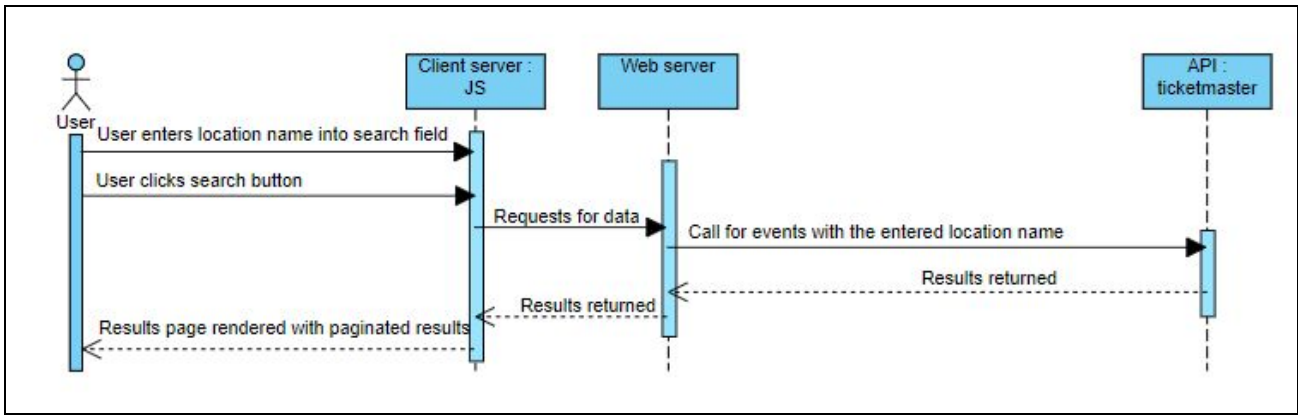
(1C) Feature: Search event by team



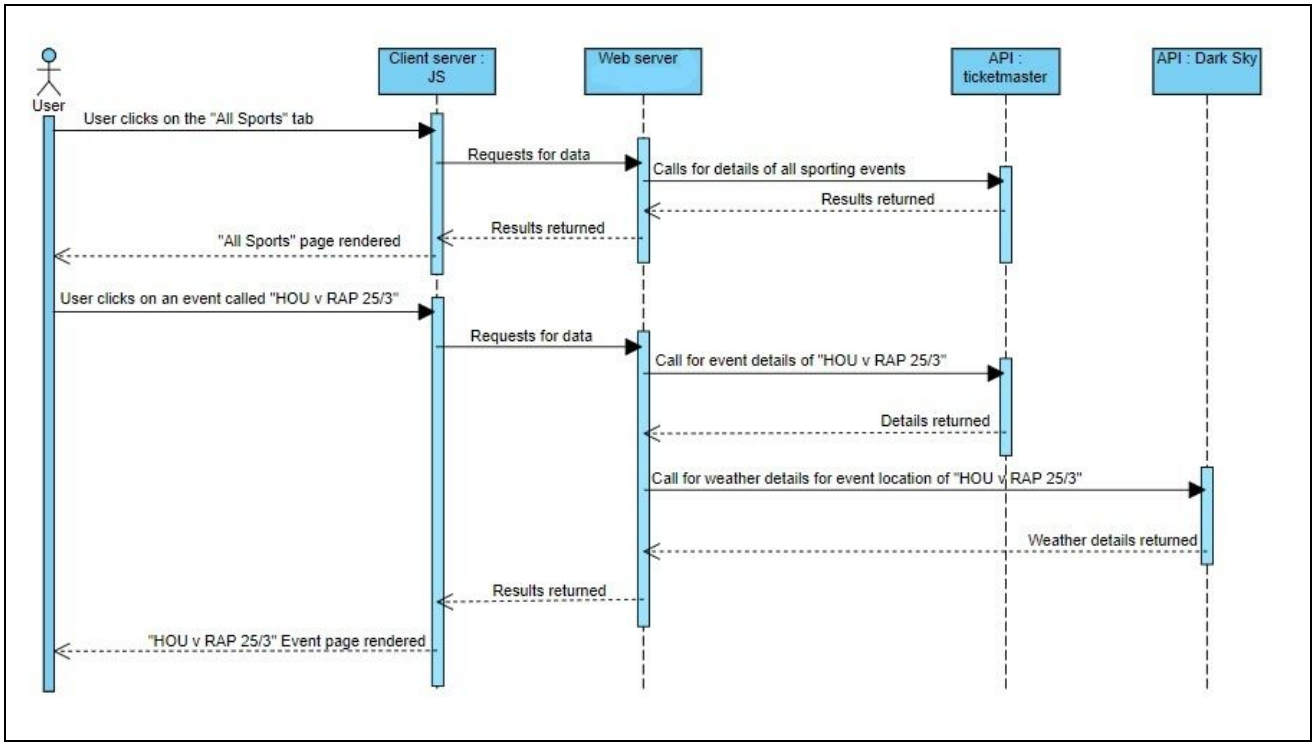
(1D) Feature: Search by genre/type



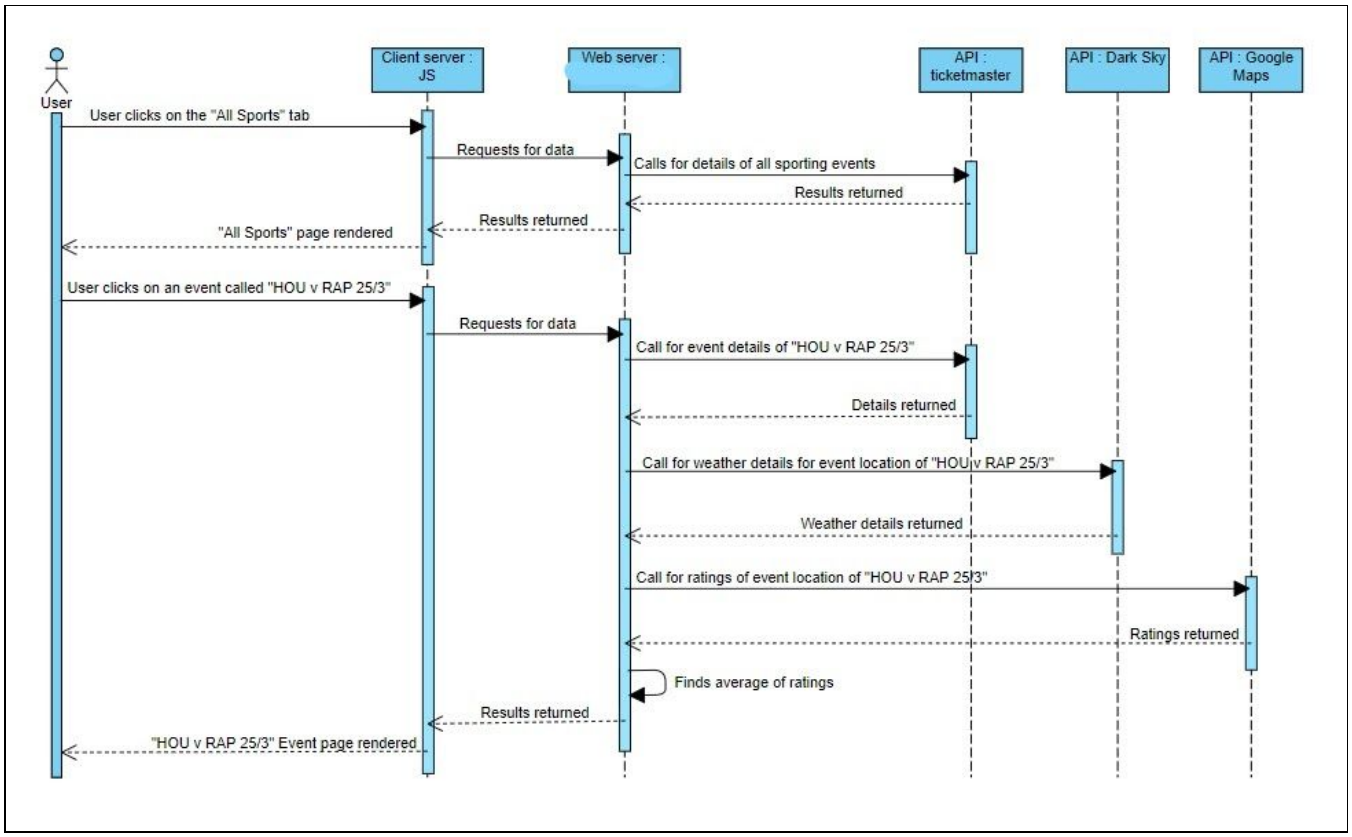
(1E) Feature: Search event by location/venue



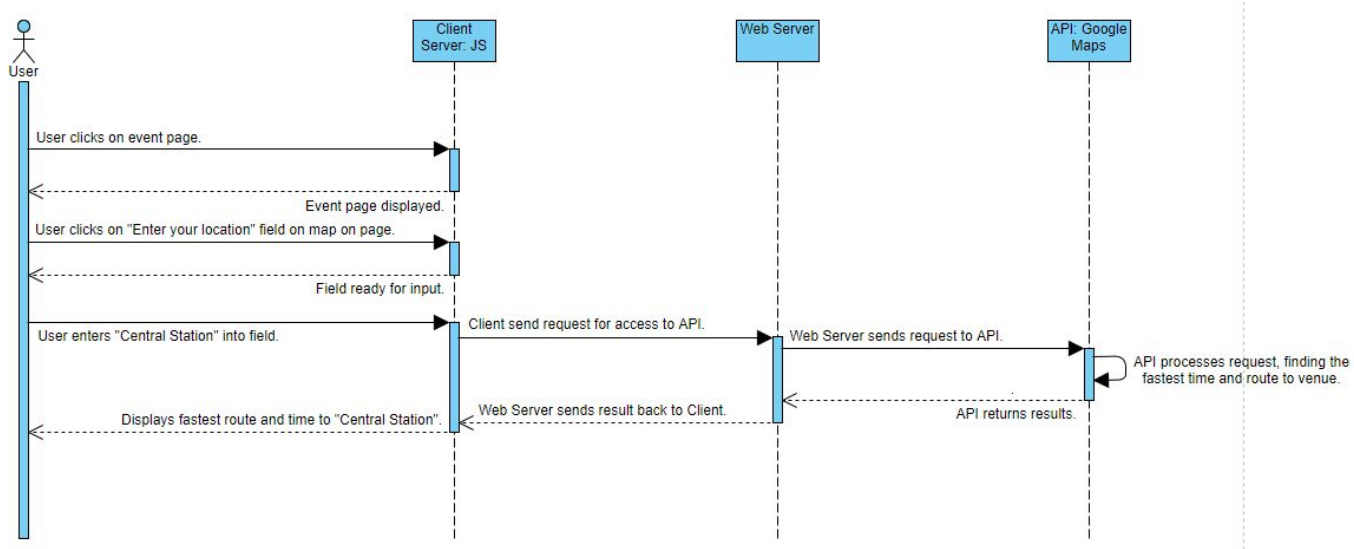
(2) Feature: Display weather for event date and location on event page



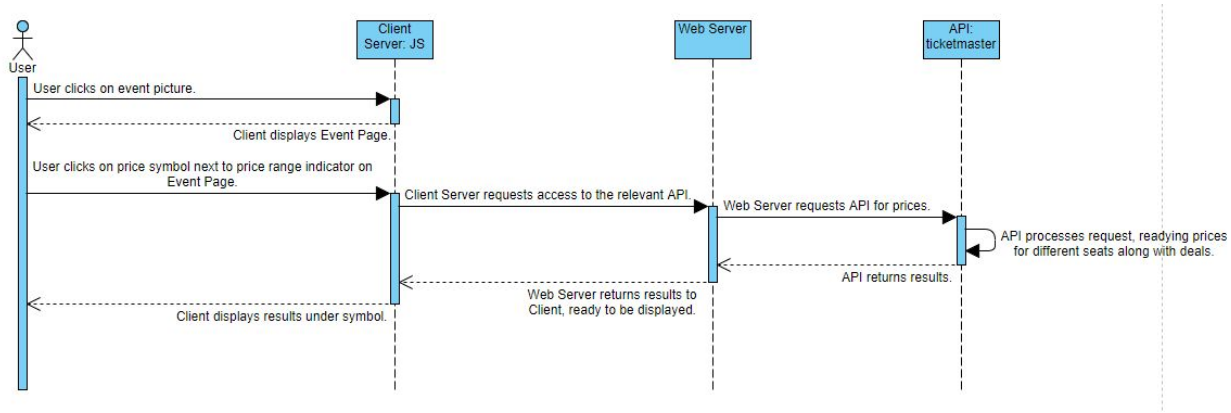
(3) Feature: Display 5 star rating system for event location on event page



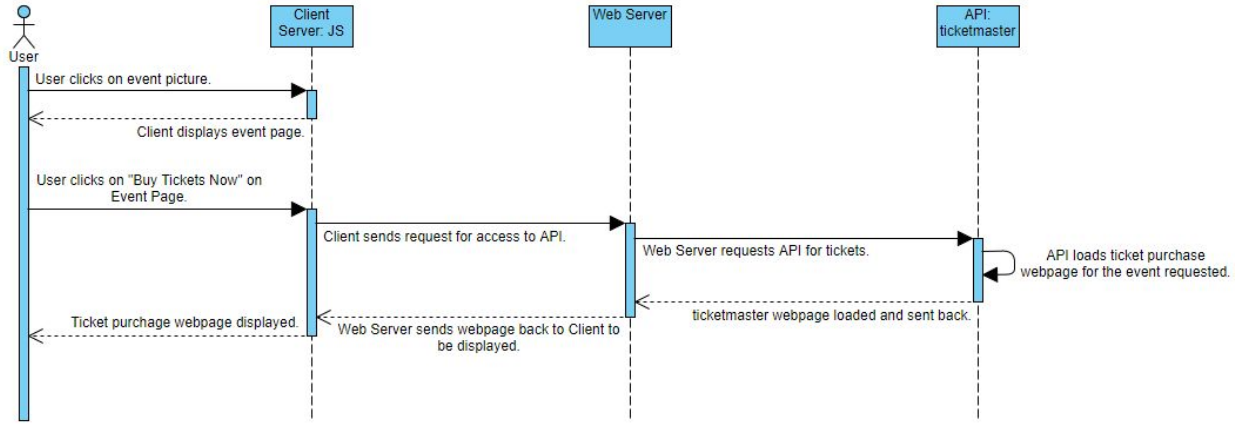
(4) Feature: Display travel directions on event date from user inputted location on event page



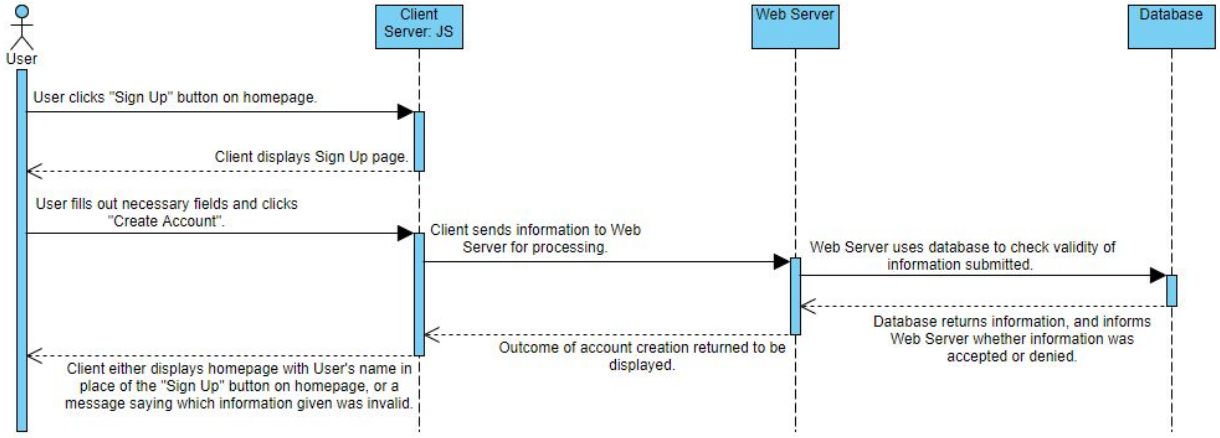
(5) **Feature**: Display ticket price ranges



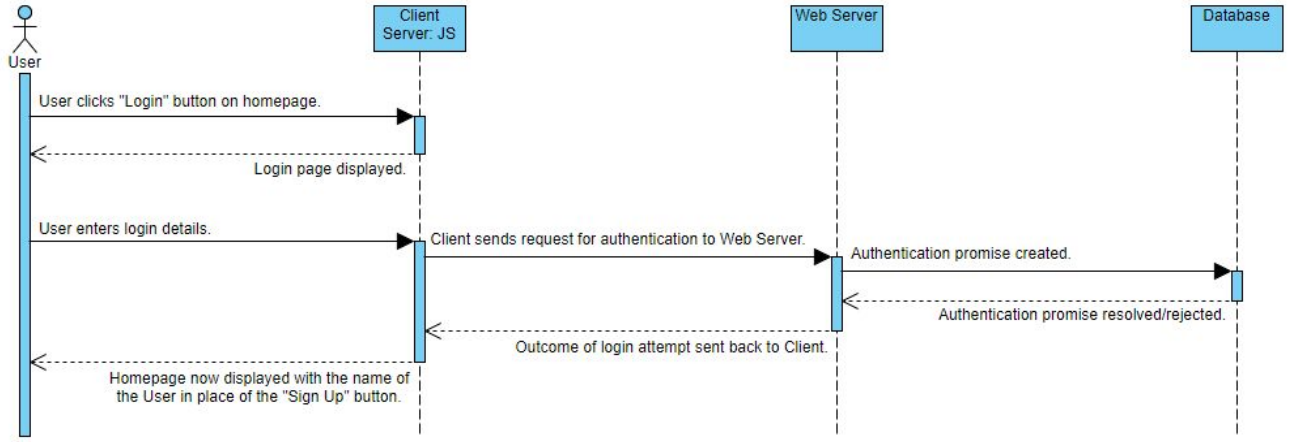
(6) **Feature**: Link to purchase tickets on external site



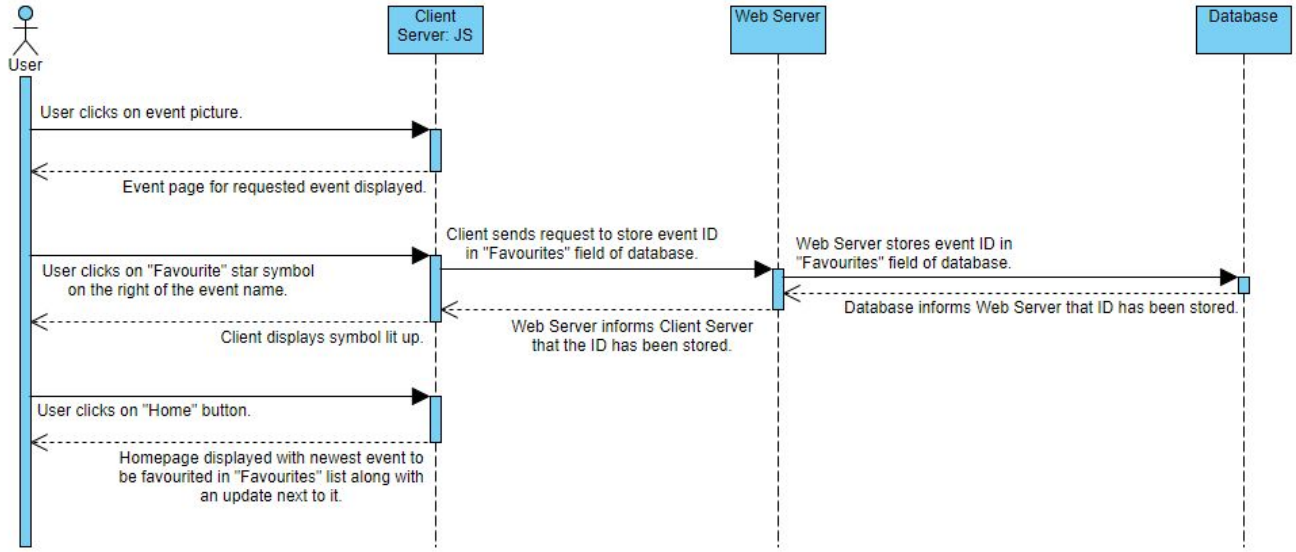
(7) **Feature**: User can register an account



(8) Feature: User can login to their registered account



(9) Feature: User can favourite events to track prices/updates when logged in



(10) Feature: User can reset password using their registered email