A sign with blood dripping on it

Description automatically generated

**Venue Scraper**

**Software Requirements Specification  
Version 2**

**Team Number:** 4

**Project Manager:** CJ Coronado

**Team Members:** James Dove, Katelyn Doyle, Raegan Durdin, Sha He, Fatima Zaid

Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary  Author(s) | Description of Version | Date Completed |
| 1 | James Dove, Katelyn Doyle, Raegan Durdin, Sha He, Fatima Zaid | Initial document submitted for CSC 154. | 10/27/2024 |
| 2 | James Dove, Katelyn Doyle, Raegan Durdin, Sha He, Fatima Zaid | Revised based on feedback from Professor Cui. | 11/3/2024 |

Review History

|  |  |  |
| --- | --- | --- |
| Reviewer | Version Reviewed | Date |
| CJ Coronado | 1 | 10/27/2024 |
| CJ Coronado | 2 | 11/3/2024 |

**Table of Contents (version 2)**

1. Introduction
   1. Project Objectives
   2. Project Scope
   3. Project Overview
2. Project Description
   1. Project Features / Functions
   2. User Stories
   3. Use Cases
   4. Project Assumptions and Dependencies
3. Project Collaboration and Documentation
4. Project Management
5. Requirements Specification
   1. Business Requirements
   2. User Requirements
   3. Functional Requirements
   4. Non-Functional Requirements
6. High-level Design
   1. Security (Required)
   2. Hardware (Required)
   3. User Experience (Required)
   4. Architecture (Required)
   5. Database (Required)
   6. Top-level Classes (Required)
   7. Data Flow and States (Required)
   8. Reports (Required)
7. **Introduction**
   1. **Project Objectives**
      1. Automate Data Collection: Develop a robust web scraper that collects concert data (e.g., band/artist name, venue, ticket price, and day of the week) from multiple venue websites.
      2. Data Management: Create a system that organizes, stores, and processes scraped data for use in analysis. This could involve exporting the data to a database or Excel sheet.
      3. Sales Prediction and Analysis: Integrate R language regression analysis into the system to predict which concerts are likely to have high ticket sales based on historical and newly scraped data.
      4. User Interface: Design a user-friendly interface that allows users to:
         1. View, manage, and edit collected concert data.
         2. Run analysis tasks to predict ticket sales for different bands and venues.
         3. Display the analysis results in an intuitive manner.
      5. Seamless Integration: Ensure smooth interaction between the front-end (UI), backend (web scraper and data processor), and analysis tools.
      6. Maximize Ticket Sales: Provide actionable insights into concert booking decisions by identifying bands and venues that perform well in terms of ticket sales.
   2. **Project Scope**
      1. Development of Multi-Venue Web Scrapers:
         1. Extend the existing web scraper to collect data from multiple concert venues.
         2. Ensure flexibility to easily add or modify scraping for new venues as needed.
         3. Data fields to be collected include band/artist, venue, ticket price, and date.
      2. Backend Data Processing:
         1. Data normalization and formatting for easy insertion into an Excel sheet or database.
         2. Implement a system to process and prepare scraped data for analysis.
      3. R Integration for Analysis:
         1. Use R for regression analysis on the data to predict ticket sales potential.
         2. Create a workflow where users can trigger R scripts to run analysis on selected data.
      4. User Interface Development:
         1. Build a front-end dashboard where users can view, manage, and analyze data.
         2. Display sales projections and analysis results in a clear, intuitive format (e.g., charts, graphs, ranked lists).
      5. System Testing and Validation:
         1. Validate that the web scraper collects accurate and complete data.
         2. Ensure the system integrates correctly with the R analysis tools.
         3. Perform performance testing to ensure the UI and data analysis processes are smooth and user-friendly.
      6. Excluded Features
         1. User Login and Access Management: No need to develop complex user management functionality.
         2. Advanced Recommendation System: Only basic filtering and sorting of data are supported, without complex AI-based recommendation algorithms.
         3. Mobile Platform Support: The project will focus on web development, without mobile platform support.
         4. Internationalization Support: The project will only support English and will not include multi-language versions.
   3. **Project Overview**
      1. This project aims to develop a web-based software solution for Shirley Road Records, a Durham-based music business, to maximize projected ticket sales for concerts. The solution will automate the data collection process from multiple concert venues, perform ticket sales analysis using regression models, and provide users with a dashboard to manage, view, and analyze the collected data.
      2. Currently, Shirley Road Records manually collects data from concerts they book and other publicly available sources. The company performs regression analysis in R to rank artists based on ticket sales potential. To streamline this process, the project will expand an existing Python web scraper and build an interface that allows the user to manage and analyze data more efficiently.
8. **Project Description**
   1. **Project Features / Functions**
      1. User Login and Access Management – User should be able to create an account, login, and be able to access settings which allows them change, update, and manage what their profile looks like.
      2. Venue Search and Filtering Functionality – User should be able to use the search functionality to find venues that works for them. You should be able to do this by typing in keywords, which refer to certain kinds of venues. Should also have a filtering functionality which allows more specific, but also more diverse output. Area, cost, seating, size, stage type, etc.
      3. Data Scraping and Automatic Updates – The scraper should then take the input from the search and filtering functionality and scrape a list of websites to gather correct information. This information is then displayed to the user via the interface. The scraper should also have the most up to date information displayed, so automatic updates should be used so the website stays consistent.
      4. Database Management – Database should also be organized and fully functional so scraper can use it to both grab information for the customer and input information to stay up to date.
      5. User Interface – Application should have a user-friendly design so that the user does not have to fight the website to access the information that they want, but it instead works for them.
      6. Backend Development and API Design – Backend development should be functional, organized, coherent, and consistent across the whole application. Comments should be used to adequately explain each part of the application, proper testing set in place, as well as error handling. The API Design itself should also follow in this manner, as well as proper documentation that should be put in place and kept up to date.
   2. **User Stories**
      1. “As a booking agent, I want an easily accessible list of venues and their performance history so that I can book adequate matches between bands and venues.
      2. “As a venue owner involved in the booking process, I want an easily accessible database of ticket performance history so that I can make educated comparisons based on surrounding and past shows in certain area.”
      3. “As a venue owner involved in ticket sales, I want a website that allows for easy ticket creation, band promotion, and ticket sales comparisons so that I can make efficient sales and promotion for upcoming shows.”
   3. **Use Case**
      1. Use Case 1: View List of Scraped Venues  
         Actor: User (Visitor)  
         Goal: View a list of venues already scraped by the system.  
         Steps:
         1. User visits the web scraper application’s homepage.
            1. System displays a list of venues that have already been scraped.
         2. User can filter the list by
            1. Venue type
            2. Location
            3. Event date
         3. System updates the venue list based on the filters selected by the User.
         4. User clicks on a specific venue to view more detailed information (e.g., address, upcoming events).
         5. System presents detailed venue information to the User.
      2. Use Case 2: Search for a Venue  
         Actor: User (Visitor)  
         Goal: Search for a specific venue from the list of scraped venues.  
         Steps:
         1. User accesses the search bar on the web scraper application.
         2. User inputs the name of a venue or keywords (e.g., "concert hall" or "outdoor space").
         3. System searches the scraped venue database for matching results.
         4. System displays the list of matching venues to the User.
         5. User clicks on a venue to view detailed information.
         6. System presents detailed information (e.g., capacity, events, contact info) about the venue.
      3. Use Case 3: System stays up-to date  
         Actor: User (Application)  
         Goal: Update system with most up to date information  
           
         Steps:
         1. Option A:
            1. Customer logs into the system.
            2. Customer views home page.
            3. Scraper runs and pulls in the most up to date information.
         2. Option B:
            1. User refreshes the page.
            2. Scraper runs and pulls in the most up to date information.
         3. Option C:
         4. After a certain amount of time has passed,
         5. Scraper runs and pulls in the most up to date information.
   4. **Project Assumptions and Dependencies**
      1. Assumptions:
         1. Team developers will be available for all portions of the developmental process
         2. All data provided will be accurate and substantial.
         3. All sprint deadlines will be met.
      2. External Dependencies:
         1. Venues providing data dependent for populating data base.
         2. Band providing data dependent for populating data base.
         3. Program/language functionality dependent for code development.
         4. The project depends on third-party websites (venue websites) for data. Changes in these sources may affect project progress.
9. **Project Collaboration and Documentation**
   1. Microsoft Teams
      1. Task tracking: project manager sends message about tasks; members clarify with project manager.
      2. Virtual meetings: meet weekly to discuss our progress, organization, and goals.
   2. Google Drive
      1. File storage: hold assignments, documentation, notes, and research for each week’s tasks.
   3. Blackboard Drive
      1. Assignment storage: information, videos, and training from instructors to reference and clarify to facilitate our work.
      2. Groups: communicate with our group members about the project.
   4. Text messaging
      1. Quick communication: updates from team lead, quick updates about scheduling and submission progress.
   5. GitHub / Git
      1. Collaborative coding: we can edit the code in separate branches and merge them.
      2. Coding version history: we can always track our previous code to reference for any changes and updates.
10. **Project Management**

We will be using the Agile Methodology. Each part of the project will be broken up into different sprints that focus on a different part of the project. An example of this could be Sprint 1, focusing on the basic creation of the website. Sprint 2 on error handling, and Step 3 can be for applying to a broader use. We will hold a weekly meeting to go over our goals and what has been accomplished in the previous week, as well as constant updates on what everyone is currently working on. This way the project can stay organized, and the group can stay focused. A tool that will be used is called Jira, it is used to help plan, organize, and help everyone stay on top of the project.

1. **Requirements Specification**
   1. **Business Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Description | MOSCOW |
| BR1 | Business must provide an application that accurately scrapes information from a list of websites, stores the information, and accurately displays it to the user. | M |
| BR2 | Business should make this application user-friendly and appealing to users with appealing features, including but not limited to easy search filtering. | S |
| BR3 | Business should have information stay up to date, so the web scraper periodically scrapes for information and updates the application as needed to stay ahead of other similar applications. | S |
| BR4 | Business could have a more specific scope of search so that the application has a larger scope of use and is not limited to one area so that lots of people can use the application. | C |

* 1. **User Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Description | MOSCOW |
| UR1 | Users must be able to create an account and log in  securely using their email and password. | M |
| UR2 | Users could be able to save venues and store them in a  storage library. | C |
| UR3 | Users should be able to search venues and filter based on  desired criteria. | S |
| UR4 | Users must be able to pull and compare performer’s  Ticket sale Data at specific venues. | M |
| UR5 | Users won’t be able to develop promotional material and  create tickets. | W |

* 1. **Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Description | MOSCOW |
| FR1 | The application must scrape data for lots of details like band, artist, venue, price, day of week. | M |

|  |  |  |
| --- | --- | --- |
| FR2 | The application must project concerts ticket sales based on regression calculations (Project Goal). | M |
| FR3 | The application must use authentication mechanisms to ensure secure user access. | M |
| FR4 | The application must show each piece of data for the user to easily view. | M |
| FR5 | The application should have permissions so some users can view the tools, and others can edit. | S |
| FR6 | The application could have audit tracking to be able to monitor any updates in the database or to the systems that can affect calculations. | C |
| FR7 | The application must support efficient data querying and filtering for sales prediction analysis. | M |
| FR8 | The application could bypass anti-scraping mechanisms to reliably collect data. | C |

* 1. **Non-Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Description | MOSCOW |
| NFR1 | The tool must be able to scrape multiple websites  simultaneously. | M |
| NFR2 | The tool must be able to manage a growing userbase should the stakeholder decide to transition the tool to a  website. | M |
| NFR3 | The interface should be user-friendly and intuitive to  navigate. | S |
| NFR4 | The program could have robust security for future use as a website, to prevent the stakeholder’s data from  being acquired by competitors. | C |

1. **High-level Design**
   1. Security (Required)
      1. Authentication and Authorization
         1. Ensures safety by having people log in before accessing the system.
         2. Two-factor authentication nearly guarantees that it’s you logging in.
         3. We will use two-factor authentication to ensure that no hacker can get in.
         4. For the logged in users, we will make sure that they’re authorized by giving them permissions and frequently checking those to make sure they’re up to date.
      2. Data Protection: Ensuring Sensitive Data Security During Transmission and Storage
         1. Data Encryption in Transit
         2. Data Masking and Tokenization
      3. Logging and Monitoring
         1. Tracking all actions made to the system (by web scrapers, APIs, users, etc)
         2. See who is contributing to the system (get usage statistics to improve it)
         3. If an error occurs, the information can be found (tied to a user, timestamp, etc)
         4. This will help us debug errors faster and eliminate security threats faster
   2. Hardware (Required)
      1. Processor Requirements
         1. The program will require data scraping and real-time calculations so a multi-core processor is preferred. An Intel i7, AMD Ryzen 5, or other equivalent processor is more than sufficient. As a website, servers will be required to handle large amounts of requests at a time.
      2. Memory Requirements:
         1. Scraping multiple websites simultaneously requires significant memory. 16GB of RAM is a good minimum, though 32GB or more is recommended if the stakeholder plans to expand their operations to scrape more websites. The RAM requirements will expand significantly as a website. 64 to 128GB is recommended based on user traffic.
      3. Storage Requirements:
         1. The program must store data about venues to perform its calculations. A 500GB SSD will be sufficient for the stakeholder's use. As a website, an NVMe SSD with at least 1TB of storage is recommended due to its fast data transfer speeds.
   3. User Experience (Required)
      1. Usability Requirements:
         1. Application will display consistent text styling, color scheme, and button styling across pages.
         2. Universal Symbols for search, filter, library, add, and login will be used to account for different linguistic backgrounds.
         3. The filter feature on the search page narrows results to reduce time spent searching through irrelevant results.
      2. User Flow and Navigation:
         1. Side Bar with Navigational links to other pages for efficient access.
         2. User Login/Sign Up Buttons
         3. Search and Library Buttons
         4. Pop Up window guidance to indicate features of Application such as;
            1. Navigating to search page
            2. Navigating to Library
            3. How to use filter feature on search page
            4. How to add/create a file in library to save venues.
            5. How to add data to an existing database.
      3. Responsiveness and Performance:
         1. Loading time should be under 2 seconds for page loading.
         2. Search results should display loading symbol and load within 30 seconds to enhance user satisfaction.
      4. Feedback and Error Handling:
         1. Success Message should appear, “Venue successfully added to file!”, when successfully adding venue to a save file.
         2. Success Message should appear, “Data successfully added to database!”, when venue data has been successfully added to database.
         3. Error messages should display for invalid inputs--” Error: Invalid Input”.
         4. Error report option should display in case of application crash--” Application has crashed. Would you like to report and share data on this crash?”.
      5. User Profiles and Personalization:
         1. Users can access their account through the account icon/page.
         2. In the account page they can edit their username, password and accessibility settings.
         3. Users can personalize their search results by favoring and unfavoriting venues. A star button will indicate favoring, and a “X” will indicate unfavored and that they do not want it to appear in future search results.
      6. User Interface Wireframe  
         A screenshot of a computer

         Description automatically generated
   4. Architecture (Required)
      1. Our project can be split into:
         1. Data collection (web scraping, APIs)
         2. Data storage (in the Excel file)
         3. Data analysis (regression models, profit calculations)
      2. Based on this splitting, the most useful architecture would be **Component-based Architecture**
         1. Individual systems (components) that interact with each other but run separately
         2. Allows you to easily improve the systems individually and run them together, too
         3. In our project, our 3 three distinct components all serve different purposes
            1. Data collection component - Collecting data from our sources

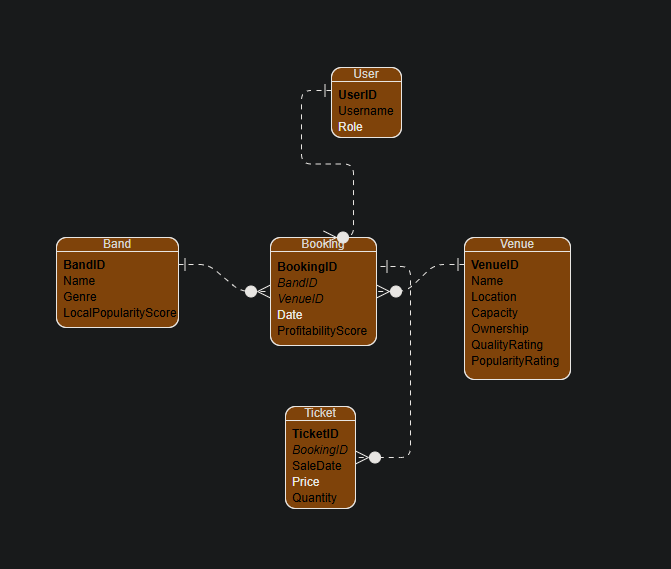
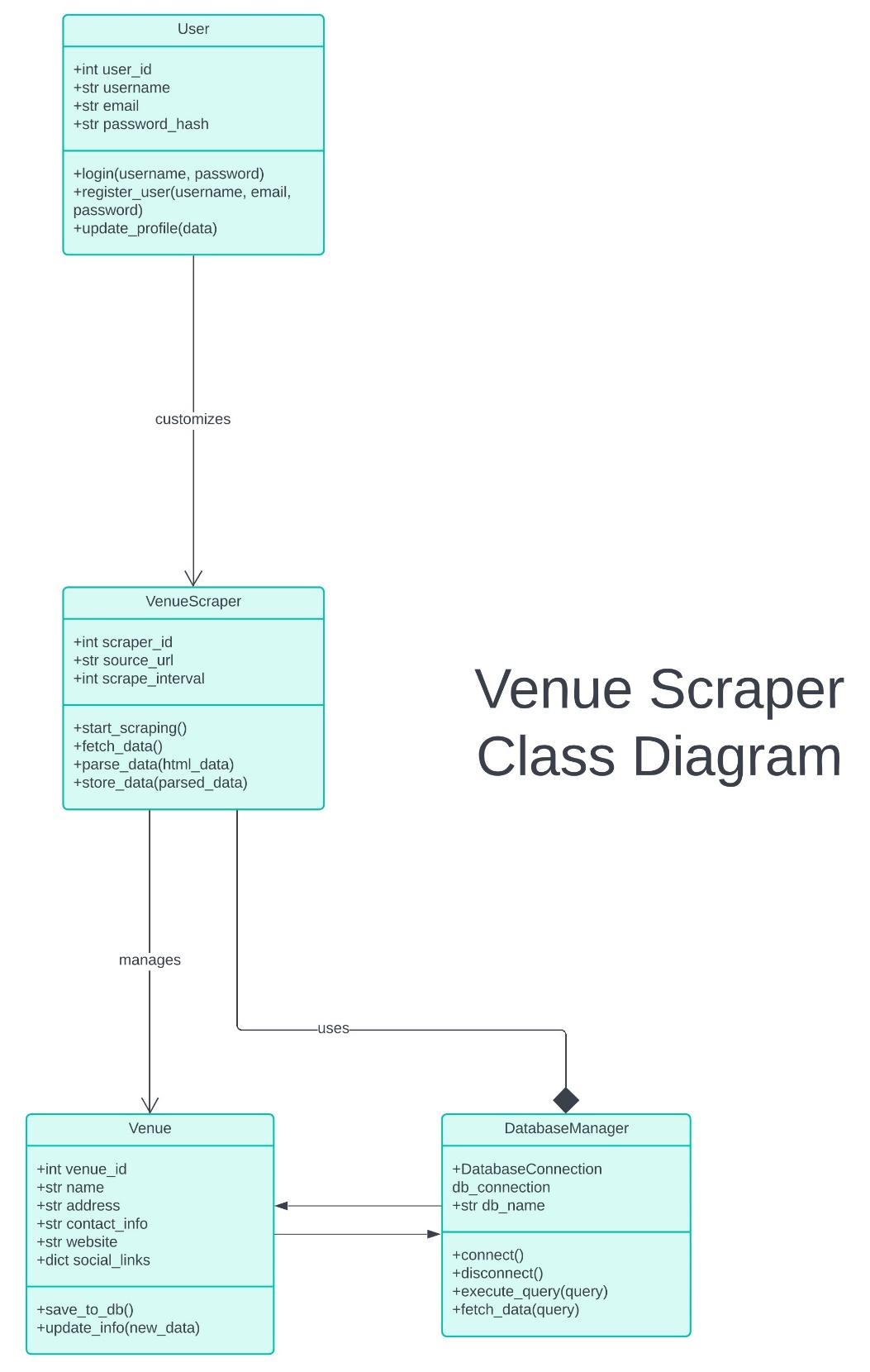
We can collect multiple sources or find most accurate ones

* + - * 1. Data storage component - Holding our data in a spreadsheet

We can query the data to easily access certain information

* + - * 1. Data analysis component - Calculates the maximum profit

We can use this to run regression models and calculations

* + - 1. These components are combined into a dataflow: Collection → Storage → Analysis
  1. Database (Required)
     1. Considering the scalability and performance requirements of the program, a cloud-based relational database management system is recommended. Amazon RDS is a good choice. It provides automatic backups, software patching, and easy scaling. It also easily integrates with other AWS services should the stakeholder take an interest in any.
     2. Below is a potential database structure.
        1. Band: Represents the bands the stakeholder is working with
           1. BandID (Primary Key)
           2. Name
           3. Genre
           4. LocalPopularityScore
        2. Venue: Represents the venues where the concerts will be held
           1. VenueID (Primary Key)
           2. Name
           3. Location
           4. Capacity
           5. Ownership
           6. QualityRating
           7. PopularityRating
        3. Booking: Represents the bookings of bands at venues
           1. BookingID (Primary Key)
           2. BandID (Foreign Key)
           3. VenueID (Foreign Key)
           4. Date
           5. ProfitabilityScore
        4. Ticket: Represents the tickets sold for each booking
           1. TicketID (Primary Key)
           2. BookingID (Foreign Key)
           3. SaleDate
           4. Price
           5. Quantity
        5. User: Represents the users of the system. Only required as a website. For personal use by the stakeholder, this is not necessary.
           1. UserID (Primary Key)
           2. Username
           3. Role (e.g., Admin, Agent, VenueOwner)
  2. Top-level Classes (Required)
     1. 
  3. Data Flow and States (Required)
     1. **Understand the Project’s Modules and Their Interactions**
        1. **Identify Modules**: Begin by identifying the primary modules or components in the project, such as:
           1. **Crawler Module**: Responsible for fetching venue and event information from external websites.
           2. **Data Storage Module**: Stores the fetched data, e.g., in a database.
           3. **User Interface (UI) Module**: Allows users to view, manage, query, or filter data.
           4. **Data Analysis Module**: Analyzes data using methods such as regression analysis to generate sales forecasts.
        2. **Interaction Workflow**: Clarify the data interactions between these modules. For example:
           1. The Crawler Module fetches data from external websites and sends it to the Data Storage Module.
           2. The User Interface Module retrieves data from the Data Storage Module and displays it to the user.
           3. Users can trigger analyses in the UI, where the Data Analysis Module processes the data and returns results.
     2. **Define Data Flow and States**
        1. **Data Flow**: Describe how data moves between different modules in the system.
        2. **States**: Define the primary states that data may encounter within the system. For example:
           1. **Data Collection State**: The Crawler Module is actively gathering data.
           2. **Data Storage State**: Data is being stored in the database or files.
           3. **Data Query State**: Users retrieve data from the database.
           4. **Data Analysis State**: The system is processing the user’s analysis request and generating a forecast.
     3. **Represent Data Flow Using a Data Flow Diagram**
        1. **Key Elements of the Data Flow Diagram**:
           1. **Entities**: The primary participants or roles within the system, such as users, the Crawler Module, and the database.
           2. **Processes**: The tasks performed within each module, such as “Data Collection,” “Data Storage,” and “Data Analysis.”
           3. **Data Stores**: Locations where data is stored, such as a database or file system.
           4. **Data Flows**: Indicate how data moves through the system. For example, a user requests analysis, and data flows from the UI to the Data Analysis Module, then returns with results.
     4. **Design the Flow and Sequence of States**
        1. **Show Data Flow and State Transitions in Order**:
           1. **Data Collection**: The Crawler Module starts, and data flows to the database.
           2. **Data Storage**: Data is stored in the database.
           3. **Data Query and View**: Users request data from the database through the UI, and data is retrieved.
           4. **Data Analysis**: Users trigger an analysis request; data flows to the Data Analysis Module for processing, which then returns a forecast.
        2. **Add Annotations**: In the diagram, add notes explaining where the data originates, where it goes, and what each state entails.
        3. A diagram of a data processing process

           Description automatically generated
  4. Reports (Required)
     1. Sales Reports Outline
        1. Reporting Capabilities:
           1. Sales Trends: Line charts and graphical visuals will be used to showcase sales trends during different parts of the year. This will allow the user to identify peak seasons for shows and identify popular venues.
           2. Performance Analysis: In association with the sales trends, the performance analysis will identify profit and attendance rate achieved by venues and their ticket sales. The venue performance analysis will also be used in combination with band performance analysis to provide evidence for decision-based making when there are multiple options of venues.
           3. Profitability: Profit analysis for each event of a venue will be reported to highlight the future optimal bookings based on the profit analysis of past events.
        2. Data Available:
           1. Ticket Sales Data:

Date of sale—Timeline to showcase the number of tickets sold between booking time to night of the show. Highlights efficiency of promotion each venue provides.

Number of tickets sold—Online Vs. In-Person.

Price per ticket—Expanded to “Price average", depending upon whether different seats cost different prices.

* + - * 1. Band Performance:

Total Profit

Attendance Rate Vs. Tickets Sold

* + - * 1. Venue Metrics:

Revenue produced by venue on average per event

Occupancy rates—Amount of people the venue can hold

Cost of hosting events—Average cost of hosting per event

* + - 1. Presentation
         1. Interactive Dashboards: Real-time visualizations of sales trends and performance metrics. Including side by side comparison between venue options.
         2. Summary Reports: Weekly or monthly PDF/Excel reports summarizing key metrics with graphic visuals comparing current metrics to previous reports.
         3. Detailed Charts and Graphs: Illustrates sales, attendance, and revenue rates.

Bar graphs

Line charts

Pie charts