**CS3354 Software Engineering**

**Final Project Deliverable 2**

Concert Booking System with Song Requests

Noah Bowman, Pranjal Ghimire, Jae Choi, Fernando Clark del Valle, Julian Jeong, Reya Dawlah, Rohit Nair

1

**Delegation of Tasks:**

Noah Bowman

Fernando Clark del Valle

Rohit Nair

Reya Dawlah

Pranjal Ghimire

Jae Choi

Julian Jeong

2

**Project Deliverable 1 Content:**

**Delegation of tasks:**

Noah Bowman - Worked on functional requirements, diagrams, and architectural design.

Fernando Clark del Valle - Worked on functional requirements and diagrams.

Rohit Nair - Worked on non-functional requirements, set up GitHub repository

Reya Dawlah - Worked on non-functional requirements

Pranjal Ghimire - Worked on non-functional requirements

Jae Choi - Worked on diagram and project scope

Julian Jeong - Worked on non-functional requirements

**Addressing feedback:** Compared to similar ticket reservation systems such as TicketMaster and SeatGeek, our application includes the additional feature of being able to request songs from the artist(s)/band(s) performing at the event. This allows the performers to tailor their performance to the fans’ requests.

**Software process model:** Within our project scope and specifications, the Waterfall model was ideal for the linear project development process that we are following. Starting with a detailed design process, we can make progress toward Development and Testing in a straightforward manner in the future. With no customer, it is difficult to implement an evolutionary model as we have no users to get feedback from.

**GitHub Repository Link:** <https://github.com/rn-utd/3354-Ctrl-Alt-Defeat>

**Functional requirements:**

1. View all upcoming concert dates
2. Open form to reserve a ticket
3. While buying a ticket, select up to 3 song preferences for performance
4. Refund ticket if bought within 24 hours
5. Allow artists to create concerts

**Non-functional requirements:**

1. Usability

The UI should be interactive and also have accessibility settings so that everyone can use the application.

1. Performance

The application should load within 3 seconds for up to 50 users at any instance.

1. Space

The website would be at most 3 mb of storage.

1. Dependability

The percentage of failure shouldn’t exceed 10% of use cases.

1. Security

The information provided by the customer will not be sold and be used only for intended purpose.

1. Environmental

The application runs 24/7 and requires a stable internet connection to function.

1. Operational

It runs on chromium based web browsers such as Google Chrome and Microsoft Edge.

1. Development

Safe coding practices should be followed when developing the code along with common agile practices. The program development will use Git as a version control system.

1. Regulatory

The application should follow WCAG accessibility standards.

1. Ethical

The application should provide transparency about user data sharing, licensing, and privacy policy.

1. Accounting

The project cost should be budgeted according to General Accepted Accounting Principles (GAAP). Purchases will be limited to the U.S only.

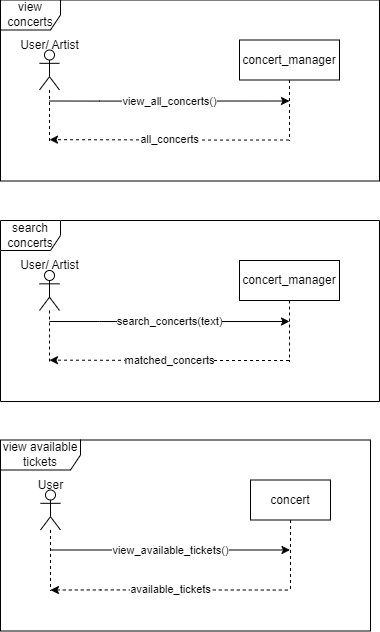
1. Safety/security  - Warning should be provided for age restricted concerts before ticket purchase.

**Use Case Diagram**

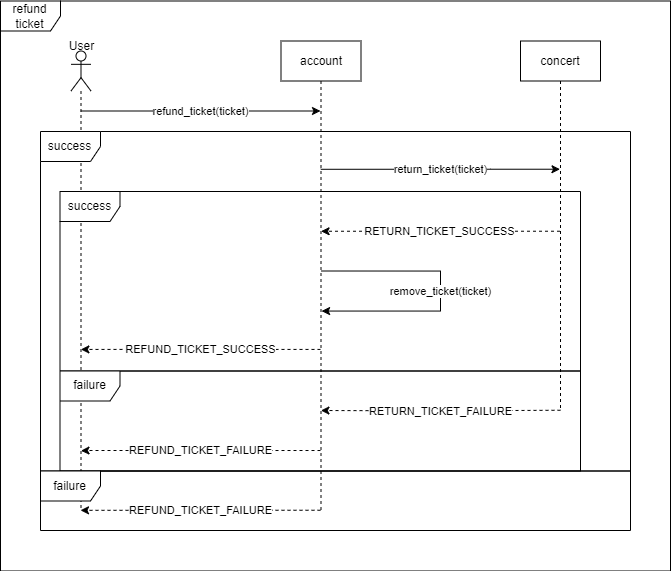
A diagram of a music concert

Description automatically generated

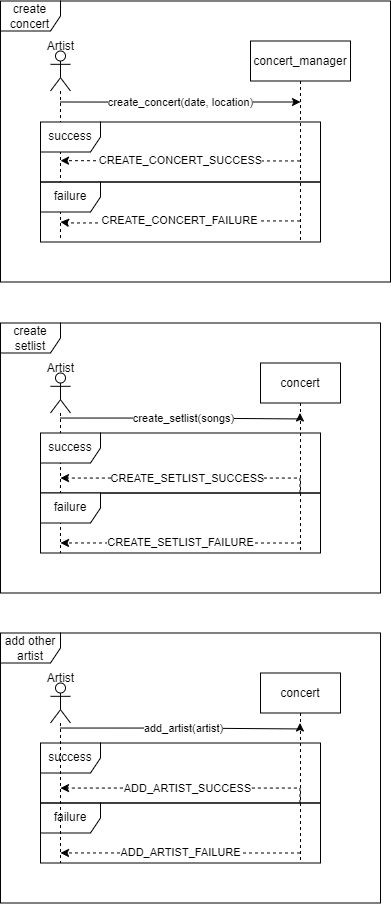
**Sequence Diagrams**



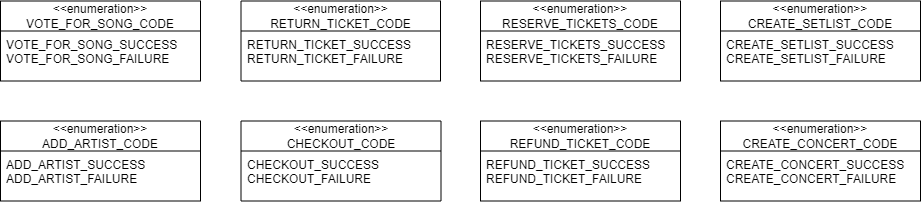
A diagram of a company

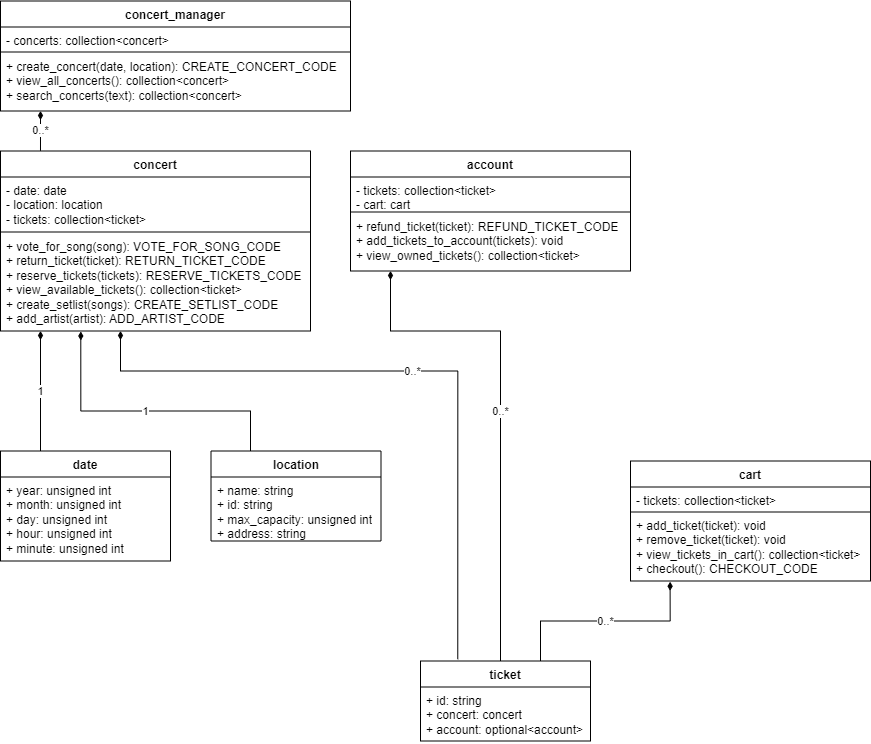
Description automatically generatedA diagram of a voting process

Description automatically generated with medium confidence

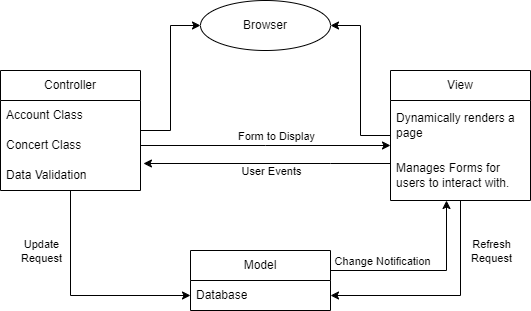


**Class diagram**





**Architectural design**



3.1

**Project Scheduling Estimation**

Start and End Date

Assuming a start date of November 15, 2023:

If weekends are not counted, with a team working 5 days a week, the end date would be approximately 3 weeks later in a 5-day workweek scenario. This would place the end date around December 6, 2023.

If weekends are counted and work is done 7 days a week, the end date would still be roughly 3 weeks from the start date, considering continuous work without breaks, which would be December 5, 2023.

Working Days Consideration

Weekends:

If weekends are not included in the schedule, then the project duration would only count the weekdays (Monday to Friday).

If weekends are included, then the project will be completed sooner, and you would have to adjust for the additional costs of weekend work if it incurs overtime rates.

Working Hours per Day

Work Hours:

The number of working hours per day directly impacts the project duration. A standard 8-hour workday is common, but this can vary.

For instance, if a standard 8-hour workday is assumed, then the total number of hours for the project would be 8 hours/day \* 5 days/week \* 3 weeks = 120 hours for the project.

If the team is working longer days, say 10 hours, then the project could potentially be completed in fewer days.

Justification

Project Duration:

The estimation of 3 weeks for project completion is based on the calculated effort of 22 person-weeks and a team size of 7.

Staffing:

The team size of 7 is justified by the estimated effort and the desire to complete the project in 3 weeks. A larger team could complete the project faster but may not be as cost-efficient.

Costs:

The project costs are estimated based on the average monthly salary of the developers and the project duration. Additional costs such as overhead, equipment, and any potential overtime must also be considered.

3.2

**Cost, Effort and Pricing Estimation**

We will be using Function Points for the cost modeling techniques.

For our program there will be following Function points,

External Inputs:

Open form to reserve a ticket (Average Complexity)

External Outputs:

Refund ticket information (Simple Complexity)

External Queries:

View all upcoming concert dates (Simple Complexity)

Submit the form to reserve the ticket (Simple Complexity)

When the artist wants to create a new concert(Simple Complexity)

New users sign up and account updates(Simple Complexity)

Internal logical files:

User account information (Average Complexity)

Concert event details (Average Complexity)

Song preference data (Average Complexity)

External interface files:

Ticket reservation (Simple Complexity)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Function Category | Count | Complexity  Simple | Complexity  Average | Complexity  Complex | Count X complexity |
| Number of user input | 1 | 3 | 4 | 6 | 4 |
| Number of user output | 1 | 4 | 5 | 7 | 4 |
| Number of user queries | 4 | 3 | 4 | 6 | 12 |
| Number of data files and tables | 3 | 7 | 10 | 15 | 30 |
| Number of external Interfaces | 1 | 5 | 7 | 10 | 5 |

GFP 55

First, enter the counts and circle the complexities in to compute the

GFP: GFP = 1X4+ 1X4 + 3X4 + 3X10+ 1X5 = 55

Now to calculate the GSC ratings, we need to answer the following questions.

Does the system require reliable backup and recovery?

Rating: 3 (average)

Justification: For a concert booking system, reliable backup and recovery are crucial because it deals with financial transactions and important user data.

Are data communications required?

Rating: 3 (average)

Justification: The system likely requires some level of data communication for processing ticket bookings and song requests.

Does the system require online data entry?

Rating: 5 (essential)

Justification: The core functionality of the system involves online data entry for booking tickets and selecting songs.

Are the master files updated online?

Rating: 3 (average)

Justification: The master files, like concert schedules and user accounts, would need to be updated in real-time as transactions occur.

Now what we finally have is,

Rating 1 (incidental): 0

Rating 2 (moderate): 0

Rating 3 (average): 3

Rating 4 (significant): 0

Rating 5 (essential): 1

The processing complexity adjustment PCA is then computed:

PCA = (Sum of GSC ratings) x 0.01 + 0.65

PCA = (1 x 5 + 3 x 3 )x 0.01 + 0.65

PCA = 0.14+0.65

PCA = 0.79

Now we compute FP, FP = GFP \* PCA = 0.79 x 55 = 43.45 FP

(Assume that the productivity of the development team is 2 function points per person-week)

Finally, the estimated effort is obtained as: E = FP/productivity = 43.45/2 = 21.725 ≈ 22 person-weeks.

If team size = 7, then project duration is: D = E / team size = 22/7 ≈ 3 weeks.

Now assuming the average cost of a developer and project management, we can say that with the average salary of a developer being at $8500 a month, Using a 7 member team, our project should cost about approximately $60000.

3.3

**Hardware Cost Estimation**

To rent a physical server $100 to $200 per month

To use a cloud server $5 to $40 per month

To purchase a physical server $1000 - $3000 [1]

3.4

**Software Cost Estimation**

BSD open source license is free to use

The All Products Pack from JetBrains IDE is $779. Since we have 7 members, it will cost $5593 to purchase IDEs for all users. [2]

3.5

**Personnel Cost Estimation**

The salary for a web developer internship in Dallas, TX is $93,838 per year or $45 per hour. [3] Since we have 7 members, the personnel cost will be about $656k per year.

4

**Test Plan**

The test plan focuses on testing the concert ticket reservation form, which is a critical component of our application. By using a unit test module called JEST[6], the tests involve retrieving information from the user and validating the format of the input before submitting the HTML form. For example, the form cannot pass the unit tests without entering a positive numerical value for the input of number of tickets. There are two test cases used. The first one uses invalid input by entering a valid first name, last name, and number of tickets but leaving the event selection blank and expects the form validation to be false. The second test case uses valid input for all fields and expects the form validation to be true.

5

**Comparison with Other Designs**

Our design is similar to Ticketmaster, SeatGeek, and StubHub – users can view upcoming events at specific locations and reserve several tickets at an event. Comparatively, our product is much simpler, as it does not scale to large volumes of users, handle payments, handle events other than concerts, or persist data [4]. What follows is a functional and non-functional comparison of our product to Ticketmaster.

Our product is functionally similar to Ticketmaster as it allows the user to view and search upcoming concerts, view available tickets at a concert, and reserve tickets at a concert. Additionally, at any time, any user can hold any number of tickets to any number of concerts.

Functionally, our product differs from Ticketmaster as it allows users to vote on songs they want to hear from performing artists at concerts. The number of votes a user can cast is linearly proportional to the number of tickets they own for the event. Our product also does not support payment methods through third-party services as Ticketmaster does.

Our product is non-functionally similar to Ticketmaster as it exposes an API allowing multiple users to interact with our system concurrently. Specifically, our API exposes concurrent services that permit users to view concerts, search for concerts, view available tickets, and reserve tickets.

Non-functionally, our product differs vastly from Ticketmaster. Our product does not scale to the degree to which Ticketmaster does. Our product is just a prototype - at any time the maximum number of users of our product will be orders of magnitude lower than that of Ticketmaster. Also, our design lacks user authorization, user authentication, and information persistence properties that Ticketmaster’s system has. Specifically, our design does not detail the usage of a database for information persistence, instead everything is persisted in memory, while designs such as Ticketmaster rely on databases for information persistence outside of program memory [5].

6

**Conclusion**

The creation of Ctrl Alt Defeat has gone extremely smoothly from brainstorming ideas to beginning implementation. Even though we are following a waterfall model for our design process. We have adopted some philosophies from an agile model. For example, our group has done a fantastic job applying our personal strengths and recognizing the strengths of others. This mindset allowed us to move faster than the traditional waterfall model and distribute the workload evenly between members without stress. Our main goal as a group was to create a ticket buying service which made the concert experience even more interactive and personal for fans. Also enabling artists to survey their fans beyond simple statistics and gain information on fan preferences and trends. To accomplish this goal, we needed to plan carefully to avoid complete redesigning or unforeseen errors in our project.

Throughout this process, we found many obstacles and constraints that we needed to overcome to produce a solid concept and begin to implement our ideas. In our original designs we had the customer control a lot more than was necessary for our final concert setlist. Customers had full control over the setlist for their specific concert. While this ensures our goal of making concerts even more interactive and personal, it also brings many issues for artists. An artist is unable to be prepared to play every song from their discography; And even if they were able to, they would not be able to perform at their best. To solve these issues, we decided to give artists priority in deciding what their base setlist would be, and then allow customers to vote on. By adding complexity, it makes artists more likely to sell tickets using our service.

**References:**

[1] E. Brinkman, "How Much Does a Server Cost For a Small Business in 2023," ServerMania, 10-Feb-2023. [Online]. Available: https://www.servermania.com/kb/articles/how-much-does-a-server-cost-for-a-small-business. [Accessed: Mar. 15, 2023].

[2] "Pricing and licensing for Windows Server 2022," Microsoft. [Online]. Available: https://www.microsoft.com/en-us/windows-server/pricing?rtc=1. [Accessed: Mar. 15, 2023].

[3] "Monthly and yearly plans with JetBrains Toolbox," JetBrains. [Online]. Available: https://www.jetbrains.com/store/#commercial. [Accessed: Mar. 15, 2023].

[4] J. Collerton, “How I would design... ticketmaster!,” Medium, https://jc1175.medium.com/how-i-would-design-ticketmaster-7062be4c2567 (accessed Nov. 16, 2023).

[5] R. Sayd, “Navigating System Design Ticketmaster,” GitHub, https://github.com/RickSayd/navigating-system-design/blob/master/designs/ticketmaster.md (accessed Nov. 16, 2023).

[6] Jest, "Tutorial: React," Jest Documentation, https://jestjs.io/docs/tutorial-react (accessed Nov. 14, 2023).