TickIt

It's good and stuff

Software Development Plan

Version: 0.3

Approval date: 4/28/2023

File Name:	Software Development Plan	
File Location:	Google Docs: CS499 Group 6/Software Development Plan	
Version Number:	0.3	
Created By:	Patrick Berzins	1/24/23
	Jayde Holbrook	1/24/23
	Van Hudson	1/24/23
	Ben Lukins	1/24/23
	Nathaniel Smith	1/24/23
Reviewed By:	Nathaniel, Ben, and Jayde	3/9/2023
	Nathaniel, Ben, Jayde, Van, Patrick	4/28/2023
Modified By:	Van Hudson	3/9/2023
	Jayde Holbrook	4/23/2023
	Ben and Van	4/27/2023
Approved By:	Patrick Berzins	1/26/23
	Jayde Holbrook	1/26/23
	Van Hudson	1/26/23
	Ben Lukins	1/26/23
	Nathaniel Smith	1/26/23
	Jayde Holbrook	3/9/2023
	Van Hudson	4/28/2023
	Jayde Holbrook	4/28/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

CDMP	Configuration and Data Management Plan
CM	Configuration Management
CSCI	Computer Software Configuration Item
QA	Quality Assurance
SDP	Software Development Plan
SEMP	Systems Engineering Management Plan
SIT	Systems Integration and Testing

1 Overview

The Software Development Plan (SDP) establishes the software development approach, methodologies, tools, and procedures to be used during the analysis, design, development, testing, integration, deployment, and maintenance of the software for Group 6's theater ticket management project. This SDP is a dynamic document and shall be updated on a periodic basis to reflect organizational changes, lessons learned, new tools, and advances in methodologies. The SDP should be a requirement for the project developers responsible for developing and submitting the SDP document for a software development effort.

1.1 *Scope*

The SDP provides the means to coordinate schedules, control resources, initiate actions, and monitor progress of the development effort. The purpose of the document is to provide a detailed plan for the use of resources, methodologies, and techniques that provide for the development of all software that comprise the product line.

1.2 *Identification*

This Software Development Plan applies to TickIt version number 0.2.

1.3 *System Overview*

TickIt is a theater ticket management software.

1.3.1 Operational Concept

TickIt is a web-based application to be used by ticket buyers and ticket sellers. Ticket buyers will use the application to purchase tickets for shows at the Civic Center Playhouse and Civic Center Concert Hall in Huntsville, Alabama. It will also be used by ticket sellers to manage ticket sales for these shows.

1.3.2 Computer Software Configuration Items

The CIs to be included in the final release of TickIt are the application (including user interface and supporting logic) and the database.

2 Reference Documents

1. Weekly Team Report, revised weekly, beginning 1/26/2023
    a. Weekly-Report-Team-6-WkXX.xlsx
2. Ticket Reservations, provided by Adam Colwell on 1/23/2023
    a. Ticket Reservations.docx
3. TickIt Test Plan

    a. Test Plan Draft.docx
4. TickIt User Guide
    a. Ticket User Guide

3 Overview of Software Development Planning

3.1 *Requirements and Development*

System and software requirements are documented in the Weekly Team Report (Reference Document 1). Software requirements are derived from the functional requirements defined in the Ticket Reservations project description (Reference Document 2) and allocated to computer software configuration items (CSCIs).

Additional system constraints are as follows:

- System must be portable.
- Users may run the program on a PC under Windows or on a Macintosh under OS X.
- Program must be robust -- Assume most users are not computer-knowledgeable. System should be able to recover from errors, particularly input mistakes.

The development process was modeled after the Scrum Agile project management system. There was a total of five sprints, each with their own development goal. These five sprints were Software Development Planning, User Stories and Requirements, Architectural Design, Preliminary GUI Design, and Final Delivery.

3.2 *Project Documentation*

The software development team will be utilizing Google products to produce and manage project documentation. All project documents will be stored in a team Google Drive folder which all members can access. Formal documentation will be written using the Google Docs online word processor. Weekly reports will be written using the Google Sheets online spreadsheet editor. All weekly reports will be considered living documents and will be updated by the end of each week of development.

3.3 *System Life Cycle*

The software development team will be using the Agile life cycle model. First, the team will determine the scope of the project and work with the client to understand the key requirements. Requirements, user stories, and epics will be defined and the team will decide which tools, frameworks, and services will be utilized for the project. Then, the team will work to implement their designs, with the goal of creating a functional product. Before the application is deployed, the quality assurance team will perform bug testing. Finally, the team will deploy the completed application.

3.4 *Schedules and Resources*

Table 1 below displays the detailed schedule of deliverables for the project, including the schedules start and completion dates, the actual start dates and completion dates, and the assigned personnel. This schedule will continue to be updated over the course of project development.

The team will collaborate to update the Team Report (Reference Document 1) every week of development. Each Sunday until April 25th, 2023, the Team Lead will submit this report to the CS499-01 course instructor. Major milestones will include the completion of the Software Development Plan (1/26), the User Stories and Requirements (2/16), the Architectural Design (3/7), the Preliminary GUI Design (3/30), and the Final delivery (4/18).

Table 1: Detailed Deliverable Schedule

Title	Scheduled Start Date	Actual Start Date	Scheduled Complete Date	Actual Complete Date	Assigned to	Verified by
Team Report Week 1	1/17/2023	1/17/2023	1/22/2023	1/22/2023	ALL	Jayde Holbrook
Team Report Week 2	1/24/2023	1/24/2023	1/29/2023	1/29/2023	ALL	Jayde Holbrook
Team Report Week 3	1/31/2023	1/31/2023	2/5/2023	2/5/2023	ALL	Jayde Holbrook
Team Report Week 4	2/7/2023	2/7/2023	2/12/2023	2/12/2023	ALL	Jayde Holbrook
Team Report Week 5	2/14/2023	2/14/2023	2/19/2023	2/19/2023	ALL	Jayde Holbrook
Team Report Week 6	2/21/2023	2/21/2023	2/26/2023	2/26/2023	ALL	Jayde Holbrook
Team Report Week 7	2/28/2023	2/28/2023	3/5/2023	3/5/2023	ALL	Jayde Holbrook
Team Report Week 8	3/7/2023	3/7/2023	3/12/2023	3/12/2023	ALL	Jayde Holbrook
Team Report Week 9	3/14/2023	3/14/2023	3/19/2023	3/26/2023	ALL	Jayde Holbrook
Team Report Week 10	3/21/2023	3/26/2023	3/26/2023	4/2/2023	ALL	Jayde Holbrook
Team Report Week 11	3/28/2023	4/3/2023	4/2/2023	4/9/2023	ALL	Jayde Holbrook
Team Report Week 12	4/4/2023	4/11/2023	4/9/2023	4/16/2023	ALL	Jayde Holbrook
Team Report Week 13	4/11/2023	4/17/2023	4/16/2023	4/23/2023	ALL	Jayde Holbrook
Team Report Week 14	4/18/2023	4/24/2023	4/23/2023	4/28/2023	ALL	Jayde

						Holbrook
SDP Submission	1/17/2023	1/23/2023	1/26/2023	1/26/2023	ALL	Jayde Holbrook
User Stories /Requirements	1/31/2023	1/31/2023	2/16/2023	2/21/2023	ALL	ALL
Architectural Design	2/21/2023	2/23/2023	3/7/2023	3/7/2023	ALL	ALL
Preliminary GUI Design	3/21/2023	3/9/2023	3/30/2023	4/6/2023	ALL	ALL
Beta Testing and Demo	4/3/2023	4/7/2023	4/20/2023	4/18/2023	ALL	ALL
Final delivery	4/20/2023	4/18/2023	4/18/2023	4/28/2023	ALL	Jayde Holbrook

3.5 *Training Requirements*

Project software engineers will be required to understand the training modules taught throughout the Spring 2023 CS499-01 course. There will be no formal, required training for project software engineers outside of course material.

Designers are expected to know or learn Typescript and ReactJS for use in developing the client-side website. Back-end software engineers are expected to know or learn how to use JSON and NodeJS. All project software engineers will be expected to know or learn how to use Jira for bug tracking and GitHub for collaboration.

If a team member needs to learn more about one of the tools mentioned above, they will be expected to find information that is no more than 5 years old. This is in place so that all educational tools are still accurate for modern versions of the tools.

4 General Software Development Activities

4.1 *Development Process*

TickIt will be developed using Agile software development. With Agile development, the development of the project will be broken down into stages called sprints. Sprints are roughly two weeks in duration. Within each sprint, team members will be assigned tasks to complete. Examples of these tasks include, but are not limited to, creating log-in pages for ticket buyers and ticket sellers, developing the base database, and enabling ticket buyers to select specific ticket seats. These tasks will be broken down from user stories, which serve as a simple description of a feature that TickIt will have.

4.2 *Development Methods*

TickIt will be created using an object-oriented methodology, especially in the JSON database. The database will be separated into the different venues, with each venue having its own list of

shows. Each show will have a similar list of objects that will represent the rows of the venue and the individual seats. A class diagram will be drafted before writing any code to ensure that the program follows object-oriented design principles.

There will be no automated OOP or testing tools used. All tests and code reviews will be conducted by team members.

4.3 *Product Standards*

All progress on the development of TickIt will be documented through team and individual weekly reports that will be submitted to the customer. It is a requirement that no code or other work will be stolen content and must be original work. It is expected that all proposed features and constraints of TickIt be implemented as best as possible.

4.4 *Critical Requirements*

There is at least one critical requirement that TickIt will have that protects the security and privacy of ticket buyers. Only ticket sellers may see, access, and export ticket buyers' personal information. No public user of TickIt will be able to access this information, as ticket sellers must sign in to a password protected page to view or change information regarding ticket buyers or event tickets.

5 Detailed Software Development Activities

5.1 *Project Planning and Oversight*

We will be developing clear plans for software development, including diagrams for the application's architectural design as well as a test plan. The Software Development Plan as a comprehensive summary of all software development planning will be reviewed by the team and updated regularly.

5.2 *Establishing a Software Development Environment*

All team members will utilize software development environments that are available for free, including Microsoft's VS Code IDE. Github will be used for version control across all team members. Jira will be used to track bugs as they come up in testing before fixes are made and pushed to Github. The NodeJS library will server-related backend and the ReactJS library will be used to develop the frontend. Additional frameworks being utilized include Bootstrap and Express.

5.3 *System Requirements Analysis*

Requirement analysis was conducted by all team members. The analysis of requirements allowed all team members to generate User Stories and Requirements for TickIt. Some of the requirements include but are not limited to, allowing ticket buyers to select their chosen seats,

allowing ticket sellers to update season ticket holder information, and allowing for ticket sellers to record purchases made at the door.

5.4 *System Design*

System requirements will be minimal due to the web basis of this application and its lightweight implementation. The web application is designed to be hosted locally on a personal machine. NodeJS and ReactJS, the key libraries being used to develop TickIt, are supported by a wide variety of devices and browsers. Other libraries and frameworks, such as Bootstrap and Express.JS, are also widely supported. The system is also designed to operate on Windows, Linux, and Macintosh machines.

From an architectural perspective, the system is mainly split into front-end and back-end parts. It is a very similar design to the MERN stack (Database, Back-end, Front-end). Traditionally, a MERN stack uses MongoDB for Database Management, but TickIt was designed to use JSON file format. The System class and router.ts files serve as the middle-man for back-end and front-end communications.

5.5 *Software Requirements Analysis*

All software requirements will only be entered once all prerequisite tasks are completed. If all prerequisite tasks are either completed or in progress, it may be entered at the discretion of the team to which the requirement is assigned. A software requirement will only be exited once all its tasks have been both completed and sufficiently tested (80% unit tested in addition to an initial successful integration test).

At the end of the project life cycle, all requirements were met with a minor exception. Requirements stated that file importing should support multiple file types, but the team was unsuccessful in supporting anything other than JSON format.

5.6 *Software Design*

TickIt's backend is to be programmed in TypeScript using NodeJS. Its frontend will be programmed in JavaScript using ReactJS. Software design for a requirement will be entered once that requirement is entered. It will be considered exited once a sufficient plan has been determined and the team agrees that development proper can begin.

A UML diagram detailing the class hierarchy and backend architectural design is shown in Figure 1. It can also be accessed at this link. Figure 2 shows the application's folder structure. It can also be accessed at this link.

Figure 1: UML Diagram

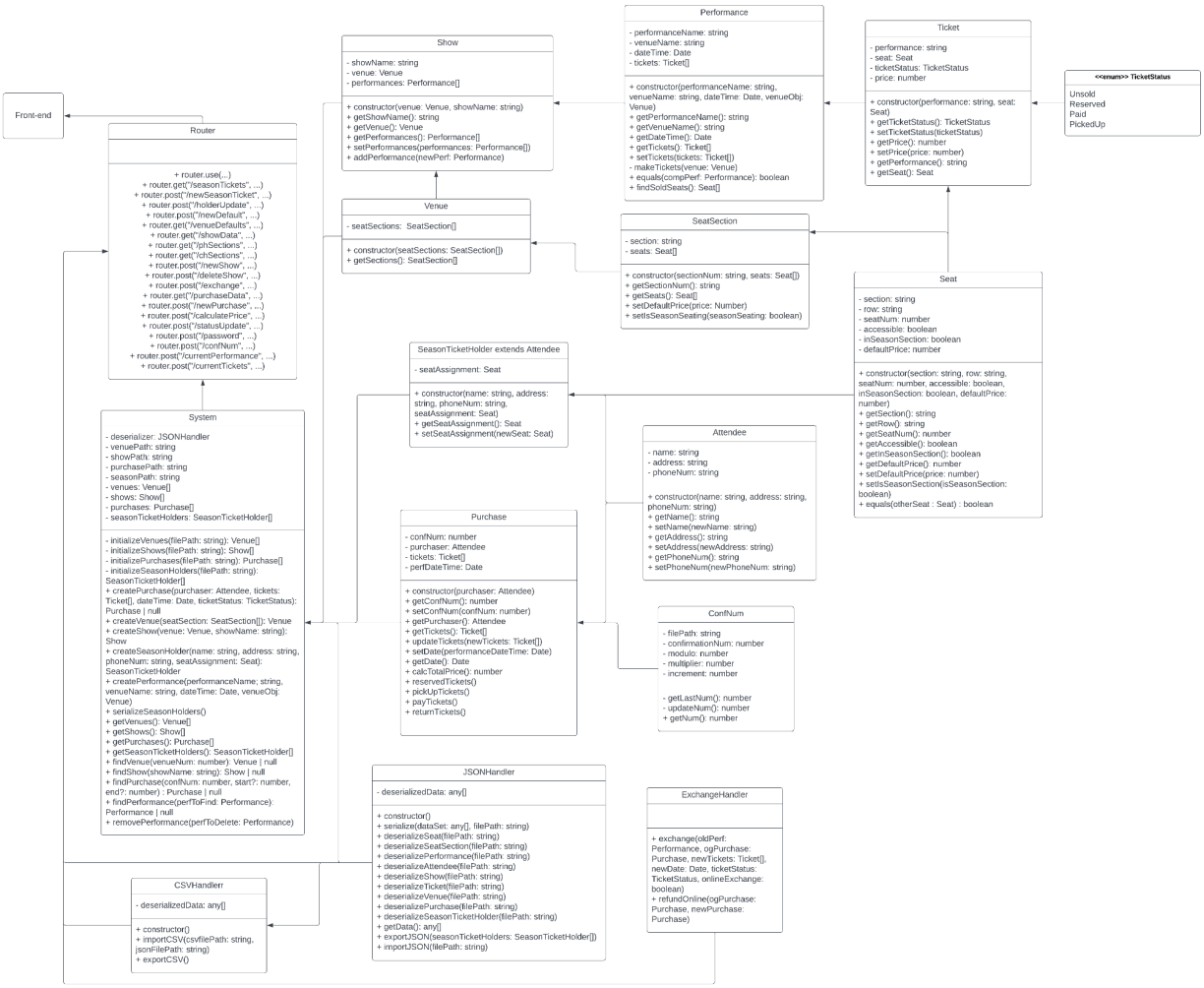

Figure 2: Folder Structure

5.7 *Software Implementation and Test/Code and Unit Test*

The Test Plan involves utilizing GitHub branches and pull requests to allow for easy integration and primary functionality testing. Upon completion of a functional piece of code, a pull request will be made, and other team members will test and approve the code functionality, while also ensuring that no existing functionality was impacted. When the project is nearing completion, detailed test cases will be created to ensure that all requirements are met and that the project is ready to deliver. Our tests should be recorded in the team's Weekly Report document and any flaws discovered will be added as Problem Reports in the same Weekly Report. This should help ensure that all code is tested properly and all issues are communicated to the team.

5.8 *Software Integration and Testing*

In addition to passing unit tests, all code should be peer reviewed and subject to integration testing. This should keep new changes from breaking old functionality and ensure units work together properly, common problems in software teams without a peer review or integration test

process. We plan to utilize Git branches to protect the working code while giving us an environment to integrate new code. These test results should also be documented in a process that is TBD.

5.9 *Preparing for Software Use*

The software will be delivered to the customer through Canvas' turn-in process. The deliverables shall include all source code files and miscellaneous documentation. Miscellaneous documentation includes, but is not limited to, team meeting notes, weekly reports, test plan, user manual, etc. The customer will also be given links to view the GitHub repository and Jira project board.

5.9.1 Preparing for Software Transition

The software was prepared for transition by all team members conducting a final clean-up of comments and unnecessary code on all source files. Rigorous testing was conducted to ensure the program was as bug-free and functional as possible. Supporting documentation was reviewed by all team members before being approved. Once all files were approved for delivery, they were placed in a folder and zipped. This zip folder was then submitted to Canvas.

The zip folder can be downloaded from Canvas and unzipped. There are specific instructions on how to run the web application in the TickIt User Manual. A Batch Script file can be used for quick and convenient boot-up on Windows. A Shell Script file does the same for Linux and Macintosh machines.

5.10 *Software Configuration Management*

Software Configuration Management was conducted via GitHub and GitHub Desktop. The GitHub repository contains all changes made to the files used in TickIt. Changes were also tracked through the team weekly report and Jira board. As team members completed tasks, the overall purpose of changes to files were summarized and tracked through the weekly report and Jira task board.

5.11 *Corrective Action*

In the event that a team member encounters a roadblock, they are expected to notify the rest of the development team as soon as possible. When a team member encounters a bug during development or during bug testing, they will create a problem report in the Team Weekly Report. Communication will be conducted via the team Discord server.

5.12 *Technical and Management Reviews*

All code will be peer-reviewed by a team member within the same role (ie. front-end code will be peer-reviewed by fellow front-end developers, back-end code will be peer-reviewed by fellow

back-end developers). Once code has been reviewed in a new GitHub branch, it may be approved and merged into the master branch.

5.13 *Other Software Development Activities*

Along with Agile development, risk management and planning will also be a key process in the development of TickIt. Additionally, security and management metrics through weekly reports will be utilized to progress development throughout the project's schedule.

Risk management, including known risks and corresponding strategies

- Team Member Drops Out/Gets Fired: 8
    - Verify that everyone would like to graduate
- COVID: 9
    - Wash your hands and stay home if sick
- Class/work conflicts: 3
    - Communicate when we are and aren't available
- Member(s) not learning required tools: 8
    - Using online resources to learn
- Technology breaks down: 8
    - Using well-known and reliable resources and redundancy of personal technology
- Platforms do not work together: 7.5
    - Proper research beforehand
- Platforms require payment to use: 8
    - Proper research beforehand. Otherwise, research alternate platforms
- Software management indicators, including indicators to be used (i.e., metrics) through weekly individual and team reports
- Security and privacy
    - Two factor authentication
    - Don't share passwords
    - Only ticket sellers may see, access, and export ticket buyers' personal information.

6 Schedules and Activity Network

The development of TickIt is planned over the span of fifteen weeks. There are five main milestones: SDP Submission, User Stories/Requirements, Architectural Design, Preliminary GUI Design, and the Final Delivery. The schedule for the development of TickIt is as seen in Table 2. More detailed schedule information is available in the Weekly Team Report.

Table 2: Schedule of Development

Item Title	Scheduled Start Date	Scheduled Completion Date
Development Sprint 1	1/12/2023	1/31/2023

Development Sprint 2	1/31/2023	2/16/2023
Development Sprint 3	2/16/2023	3/9/2023
Development Sprint 4	3/9/2023	3/30/2023
Development Sprint 5	3/30/2023	4/20/2023
SDP Submission	1/17/2023	1/26/2023
User Stories/Requirements	1/31/2023	2/16/2023
Architectural Design	2/21/2023	3/7/2023
Preliminary GUI Design	3/21/2023	3/30/2023
Beta Testing	4/1/2023	4/18/2023
Final Delivery	4/20/2023	4/28/2023

7 Program Organization and Resources

Our team will be organized according to the below Table 3. As the project progresses, Patrick and Van will work primarily on front-end development and UI design. In turn, Jayde, Ben, and Nathaniel will focus on back-end development and database management.

Table 3: Team Roles

Patrick Berzins	GUI Lead, Frontend Developer
Jayde Holbrook	Team Lead, Backend Developer
Van Hudson	Technical Writer, Frontend Developer
Ben Lukins	Test Lead, Backend Developer
Nathaniel Smith	Technical Lead, Backend Developer

8 User Definitions

Each ***seat*** is occupied by a ***ticket-holder.*** A ticket may be either purchased by the holder or received as a complimentary unpaid ticket (a "comp" ticket). A ***show*** or ***production*** is an entire run of one or more ***performances*** of the same theatrical work.

A ***season*** is a series of productions, usually covering a period from August to June of the following year. A ***season ticket*** is a purchase made for attending one performance of each show in the season, often at a discount from the sum of the individual tickets. A season ticket will be for the same night of the week and the same seat for each production. Most organizations derive a major portion of their revenue from season tickets.

A ***section*** is a seating area of a theater venue, usually comprising a group of seats. A ***loge*** is a section in the rear of a theater. A ***balcony*** is a seating area usually above and behind most other seats. Any section may be closed off or otherwise marked as unavailable for a particular show and/or individual performance.

A ***ticket buyer*** is any user of the system that may purchase a regular or season ticket to any performance but has not done so yet. A ***ticket seller*** is a user that is a volunteer or part of the organization that is hosting a show or production. Ticket sellers have a special sign-in page to protect sensitive personal information of ticket-holders.

DOCUMENT REVISION HISTORY			
Version Number	**Approved Date**	**Description of Change(s)**	**Created/ Modified By**
0.1	1/26/2023	Initial draft of the SDP	All team members
0.2	3/9/2023	Updates after Architectural Design sprint	Van Hudson
0.3	4/28/2023	Preparation for final submission	All team members