Absolute Cinemas

Software Requirements Specification

Version 3.0

9/18/25

Group 3
Renz Ryan Santillana
Luis Meza
Fabian Llamas

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Fall 2023

Revision History

Date	Description	Author	Comments

9/18/25	Version 0.0	Group 3	First draft for Asst 1
9/25/25	Version 1.0	Group 3	Final draft for Asst 1
10/9/25	Version 2.0	Group 3	Final draft for Asst 2
10/23/25	Version 3.0	Group 3	Final draft for Asst 3

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of "Absolute Cinemas" is to create a simple and reliable system for purchasing movie tickets online and through a mobile application and website implementation. The system will allow users to browse movie listings, view schedules, and securely purchase tickets. The SRS will guide the design and implementation process by describing functional and non-functional requirements.

1.2 Scope

(1) Absolute Cinemas will be the software product to be produced. This encompasses the mobile app versions to be released on major operating systems and the development of https://AbsoluteCinemas.com for supported browsers.

(2) Absolute Cinemas Software will grant users access to ticketing features for movies and digital media. These features will include:

- Browsing, Searching, and Previewing.
- Selection of tickets based on location, date, and time.
- Reservation of in-theater seating for a selected number of tickets.
- Confirm and Purchase selected ticket products.

This software will include general features unrelated to digital media, such as Sign-up and Login to user accounts, options to use Membership rewards, and caching of unfinished purchase progress.

(3) The application of Absolute Cinemas software catalogs and displays options for digital media to users. This provides users with benefits prior to visiting physical locations.

(a) The main objective of this software is to provide users with up-to-date information on tickets and seating, presenting benefits for users to encourage their purchase ahead of time.

(b) This software will provide purchase capabilities up to 1 hour before show times when available at the user's location.

(c) This software's goal is to effectively and efficiently process transactions from users to provide ticketing information.

1.3 Definitions, Acronyms, and Abbreviations

- UI - User Interface
- DBMS - Database Management System
- API - Application Programming Interface
- QR Code - Quick Response Code used for ticket scanning
- 2FA - Two-Factor Authentication
- SRS - Software Requirement Specification

1.4 References

The references are:

- Absolute Cinemas Structural Model
- Absolute Cinemas Software and Electronic Model
- Absolute Cinemas Database Management System Model
- Absolute Cinemas Application Programming Interface Model
- Absolute Cinemas Vision Draft

1.5 Overview

From this point onwards, this SRS document will cover two major sections in the form of General Description and Specific Requirements.
As an overview, the General Description will inform the reader of the aspects of Absolute Cinemas software that can be described on paper. This includes the perspective from which the software is designed, the functionality of each software, the intended user base, and the constraints and assumptions accounted for. The Specific Requirements describe the data requirements needed to account for both Functional and Non-Functional Requirements that comprise Absolute Cinemas.

2. General Description

2.1 Product Perspective

Absolute Cinemas is an independent software product that will connect with existing theater hardware and software. This software is dependent on systems such as scanners, payment processors, computers, and operating systems. Internet access is necessary for cloud-based servers to handle reservations, payments, and ticket verification.

2.2 Product Functions

The functions of all products will be the following:

1. Search and display movies with showtimes.
    a. Display movies based on user criteria and search terms.
    b. If no criteria are specified, display a default state.
2. Provide seat selection and availability updates.
    a. Allow for the selection of multiple seats.
3. Support secure ticket purchasing using multiple payment methods.
4. Generate and deliver digital tickets through email or app notifications.
    a. Delivery occurs within 1 minute of purchase.
5. Maintain customer profiles with order history preferences.

2.3 User Characteristics

1. The primary users are customers between the ages of 13-65 with basic smartphone or computer knowledge.
    a. Users will intend to browse, review, and purchase tickets from the application.
2. Secondary users include theater staff who will scan tickets and manage schedules.

    a. Secondary users will be allowed the basic features of primary users.
    b. Users can review customer and primary user information upon request.
3. 3. All users are expected to have internet access and a modern browser or mobile device.

2.4 General Constraints

1. The application must run on Android, iOS, and web browsers.
    a. This application is subject to updates based on changing mobile platforms and the few supported browsers.
2. The system depends on third-party payment gateways.
    a. This application must always support at least one payment method at all times.
    b. Visa and Mastercard debit types will take priority among payment methods.
3. Movie data must be updated daily from theater management systems.

2.5 Assumptions and Dependencies

The dependencies of this software can be in one of two categories: First-Party and Third-Party. Assumptions for the success of this hardware will also be grouped in this section as they become relevant.
First-Party Dependencies are delegated to the appropriate physical location and property of the customer. Such dependencies are:

- Theaters will provide accurate movie schedules and seat availability.
- Theaters will maintain access to the ticketing database to validate tickets upon use.

Third-party dependencies are taken care of by partnered services and primary users.

- Users will have stable internet connectivity.
- Payment providers will maintain secure and reliable services.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interfaces are defined by the devices accessing the application. Accounting for these differences, mobile devices will implement a different interface from web browsers.

1. Mobile Devices
    a. The interface will provide simple navigation for browsing movies and purchasing tickets.
    b. Mobile and web versions will use a consistent design on mobile devices.
    c. Mobile app versions will display simple notifications in the app and push notifications when enabled.
    d. Customers will view digital QR codes for admission.
2. Web Browser
    a. Interface optimized for web browsers.
    b. Prompt users for desktop notifications.

3.1.2 Hardware Interfaces

- Ticket scanners must support QR code recognition
- Payment terminals must integrate with the theater’s financial system

3.1.3 Software Interfaces

- The system will connect with third-party payment APIs
- It will use the theater's scheduling database for showtime updates
- It will integrate with email/SMS gateways for ticket confirmations.

3.1.4 Communications Interfaces

- All communication between client devices and the server will use HTTPS
- Internal server communication will use TCP/IP protocols

3.2 Functional Requirements

3.2.1 Feature #1: Search

3.2.1.1 Introduction: This feature allows the user to type in criteria for a search in the movie database. Search will sort by similarity and exact matches.
3.2.1.2 Inputs: user keywords or filter (title, genre, time).
3.2.1.3 Processing: query database for matches.
3.2.1.4 Outputs: list of movies with details.
3.2.1.5 Error Handling: display "no results found" if no match exists.

3.2.2 Feature #2: Payment Processing

3.2.2.1 Introduction: Payment Processing is a feature initiated by the user after selecting to check out. Payment processing connects the user with third-party payment processing services.
3.2.2.2 Inputs: Confirm payment method used for purchase.
3.2.2.3 Processing: Verify the availability of seats in a showing through UX.
3.2.2.4 Output: Confirm payment through email with a QR code attached for check-in.
3.2.2.5 Error Handling: Reject payment if the selected seats are already occupied and or an invalid payment method.

3.3 Use Cases

3.3.1 Use Case #1:

Actor: Customer 1
Description: The customer opens the application and enters a movie title in the search bar. The application displays search results and available showtimes to the user. The customer watches a preview of the movie and selects a desired showtime.

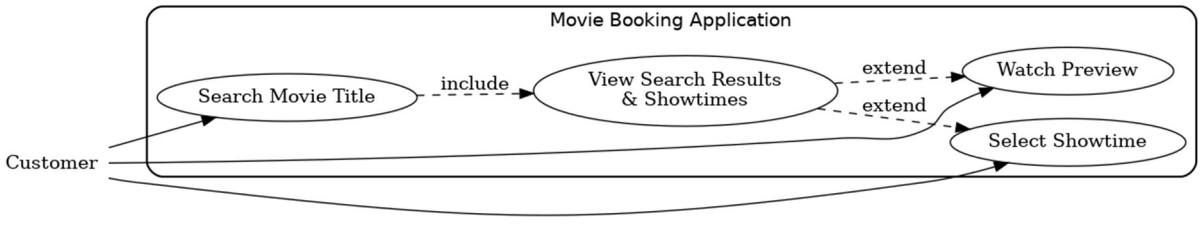


Figure 1

3.3.2 Use Case #2:

Actor: Customer 2
Description: The customer enters a valid amount of available seats to pay for with the appropriate payment information entered by the customer, and will receive a payment confirmation through email once the purchase is processed and verified.

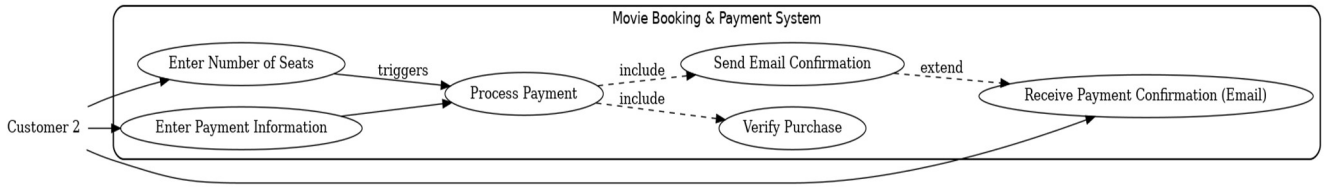


Figure 2

3.3.3 Use Case #3:

Actor: Accessibility Customer
Description: The customer filters a search for wheelchair accessible seats, application displays showtimes with handicap seating still available, customer picks seats for the desired showtime, customer is prompted to acknowledge handicap seating policies.

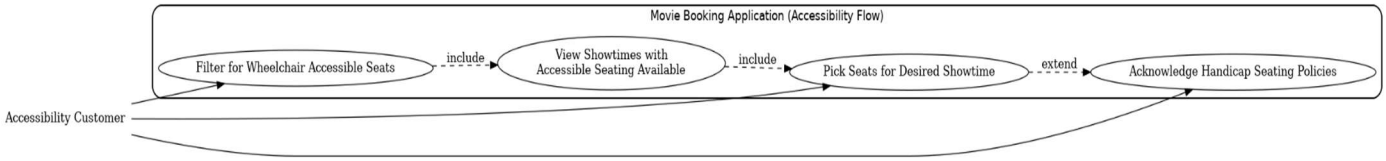


Figure 3

3.4 Classes / Objects

3.4.1 Object #1: Movie

3.4.1.1 Attributes: title, genre, duration, rating, showtimes
3.4.1.2 Functions: displayInfo(), getSchedule(), getLocation()
Reference to functional requirements and/or use cases

3.4.2 Object #2: Ticket

3.4.2.1 Attributes: ticketID, movie, seat, showtime, QRCode
3.4.2.2 Functions: generateQRCode(), validate()

3.4.3 Object #3: Seat

3.4.3.1 Attributes: rowNum, seatNum, accessible, availability
3.4.3.2 Functions: getNumber(), getRow(), getAvailability(), isAccessable()

3.4.4 Object #4: Account

3.4.4.1 Attributes: userId, name, email, phone, status
3.4.4.2 Functions: userLogin(), getOrder()

3.4.5 Object #5: Cart

3.4.5.1 Attributes: numItems, total
3.4.5.2 Functions: add(), remove(), updateAmount(), checkout()

3.4.6 Class #1: User

3.4.6.1 Attributes: location, dateAccess
3.4.6.2 Functions: guest()

3.5 Non-Functional Requirements

3.5.1 Performance

The application will be considered a light workload for all supported devices. Focused on delivering through mobile devices, the application will not be memory-intensive or computationally complex. The website version of the software will be consistent with the performance requirements of the mobile application. Therefore, performance requirements are as follows:

- 95% of search queries shall return results within 2 seconds.
- Filtering search results will occur automatically during a search.
- Ticket confirmation shall be generated within 10 seconds after payment.
- Generating QR codes occurs within seconds of request.

3.5.2 Reliability

The movie database server will be responsible for maintaining reliability across all on-site servers. Redundancy is also planned within servers in the form of data backups and uninterruptible power supplies to deploy when needed. This implementation will allow the application to maintain these features:

- System uptime must be at least 99.5%
- Daily data backups will be maintained through a separate database with cloud transfer.

3.5.3 Availability

- The system must be accessible 24/7 with minimal planned downtime; such planned downtimes must be off-hours.

3.5.4 Security

- All sensitive data shall be encrypted during transfer and storage.
- Users must log in with a password and optional Two-Factor Authentication(2FA).
- All passwords have an expiration date in a month from password creation.
- New passwords must be unique and cannot be old passwords used in the past.
- Two-Factor Authentication will be sent through mobile notifications.

3.5.5 Maintainability

- Code shall follow a modular design for easier updates.
- Commentation in classes is a requirement to avoid confusion when accessing code.
- Documentation will be maintained for developers.
- Senior programmers must create a master key for the program to be created and secured through a physical or digital safe.

3.5.6 Portability

- The application must run on Android, iOS, and major browsers (Chrome, Firefox, Safari, Edge, and Brave).
- Database and server components should be deployable on cloud platforms such as AWS or Azure.

3.6 Inverse Requirements

To prevent any misunderstandings, this software is designed not to include specific features. The reason for excluding them can range from limitations and constraints to mitigating liability. The inverse requirements are as follows:

- Storing financial and payment information.
    - This information will not be stored either locally or remotely between subsequent visits for any single user.
    - Software will state the responsibility of the user to provide information at their own risk
    - This does not include personal identifying information that users provide on their profiles.
- Software will not provide the option of purchasing concessions or food items within any of its applications.
    - Software will include a disclaimer of this policy based on location.

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitations, etc., that will impact this software project.

- Industry & Legal standards:
    - Payment flows must comply with the Payment Card Industry Data Security Standard (PCI DSS), and they cannot store any credit card information
    - User privacy is protected; it cannot be released to any third party.
    - Web/app accessibility
    - QR ticket code scanners can read them.
- Company policies:
    - No storing of payment credentials
    - Mandatory code reviews and issue tracking before releases
    - Error logs must exclude sensitive fields (e.g., full names + emails together)

- Platform rules:
    - iOS App Store and Google Play policies
    - Browser support baseline from Chrome, Safari, Firefox, Edge, and Brave

- Hardware limits:
    - Ticket scanners in theaters must read QR codes printed on standard thermal paper and on mobile screens at typical brightness

- Operational constraints:
    - All external traffic over HTTPS
    - Backend changes must be backward-compatible with the mobile app for at least one minor release
    - Peak-time protection: rate limiting and queueing around showtime drops

3.8 Logical Database Requirements

- Database usage Required: needed for movies, showtimes, seats, orders, accounts, and audit logs.
- Core entities(logical):
    - Movies(title, genre, duration, rating)
    - Showtime(movie_id, theater_id, start_time, auditorium)
    - Seat(showtime_id, row, number, accessible, status)
    - Order(order_id, user_id, total, status, created_at)
    - Ticket(order_id, seat_id, qr_token, validated_at)
    - User(user_id, email, phone, password_hash, status)

- Data format & integrity:
    - Unique constraints on (showtime_id, row, number) and on qr_token
    - ACID transaction for "reserve seat -> pay -> issue ticket" so seats don't double-book
    - Soft deletes (status flags) for Orders and Users to preserve history.

- Storage & performance:
    - Indexes on movie_id, start_time, theater_id, and seat availability flags.
    - Partition showtimes/tickets by month to keep queries fast.
    - Limit attachment size for marketing images and store references only

- Retention & Compliance:
    - Order and ticket records are kept for 24 months for reporting/chargeback, and audit logs are kept for 12 months
    - PII minimization: only necessary profile field

- Data integrity check
    - Daily reconciliation job comparing seat maps vs. ticket counts
    - QR tokens are short-lived for offline validation and revoked after the first scan

- Backup & recovery:
    - Daily full backups and hourly incrementals
    - Encrypted any sensitive information

3.9 Other Requirements

- Localization: Start with en-US, design copy, and format the dates or currency
- Observability: Centralized log, metrics, and alerting on SLA breaches.
- Support & ops: Status page for incident, maintenance windows scheduled during overnight, local time, and beta testing.
- Security extras: 2FA for users and mandatory for theater staff accounts
- Testing: Automated unit/integration tests for checkout path, synthetic "buy ticket" probes in production to catch failures early

4. Analysis Models

4.1 Class Diagram

The 4.1 class diagram is shown below. This diagram includes one node for every class implemented in this application and its relationship to objects and each other.

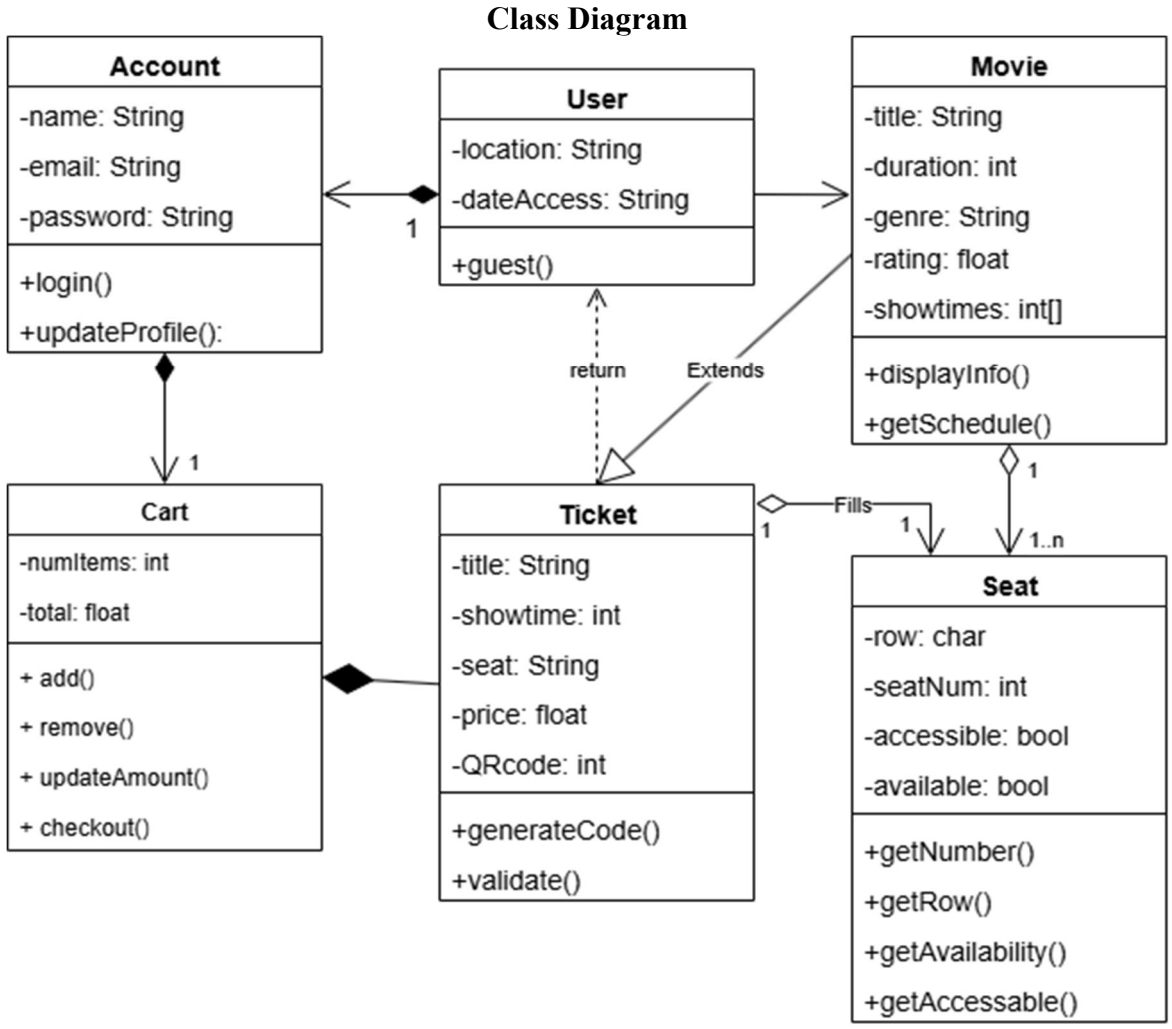


Figure 4

The purpose of this class diagram is to provide a visual representation of the user experience from start to finish. The diagram flow begins with an instance of the User class(3.4.6) and the chain of relationships that compose the Account, Cart, and Ticket objects(3.4). It is important to note that only user input will access a Movie object and compose a Ticket object from the Cart.

4.2 Data Flow Diagram (Context Diagram)

The Data Flow Diagram (4.2) is shown below. The Data Flow Diagram describes the movement of both user-input data and automatic data transfers.

Data Flow Context Diagram

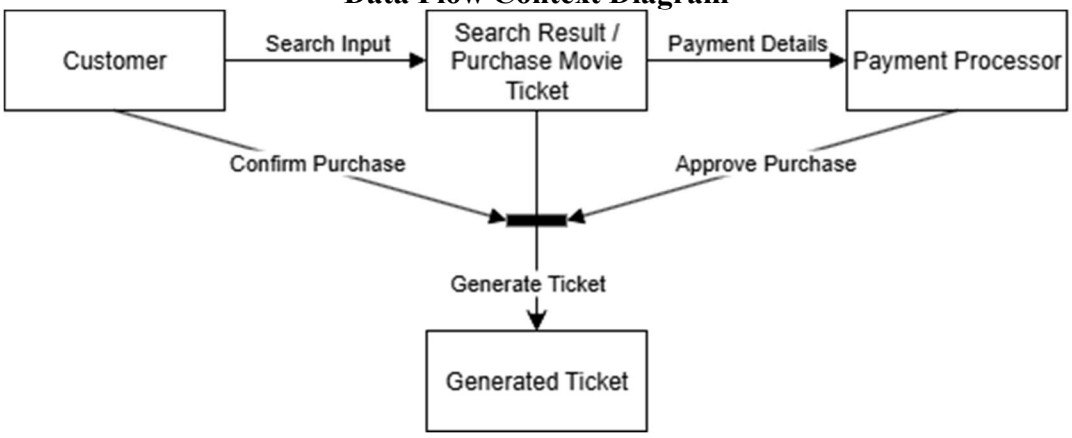


Figure 5

The purpose of this diagram is to show the context in which data flows from the user to First-party and Third-party systems. Finally, data flows back to the end-user with deliverable information. In this case, the user will interface with the application, and information must be inputted for First-Party and Third-party systems to return validation, and a ticket can be generated back to the user.

4.3 State-Transition Diagrams (STD)

The State Transition Diagram (4.3) is shown below. The State-Transition Diagrams describe the movement of users. The user went to the website and started browsing for a movie once the user has selected a movie. Then the user chooses a showtime once the user has chosen showtime. Then the user chooses a seat once the user has chosen a seat. Then the user proceeds with the payment, the app or the website processes the payment, and determines whether the user has sufficient or insufficient funds on the card or payments. Once the payment processor determines the funds, it will generate a QR code or ticket. Once the ticket has been scanned, the user is no longer able to rescan the same QR code or ticket.

State Transition Diagram

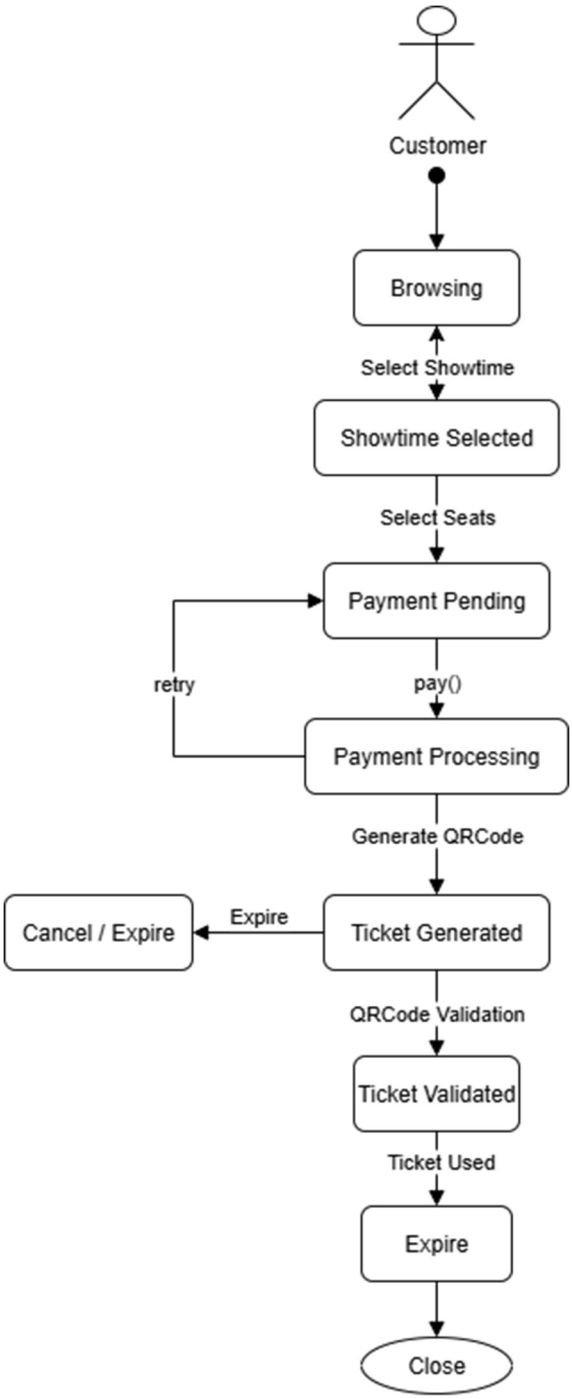


Figure 6

4.4 Software Development Timeline

This is an overview of the estimated Software Development Timeline, projecting future time costs according to the five development stages. Requirement Gathering, Resource Planning,

Software Development, Testing, Verification and Validation are the five stages this software is expected to complete within the given timeline.

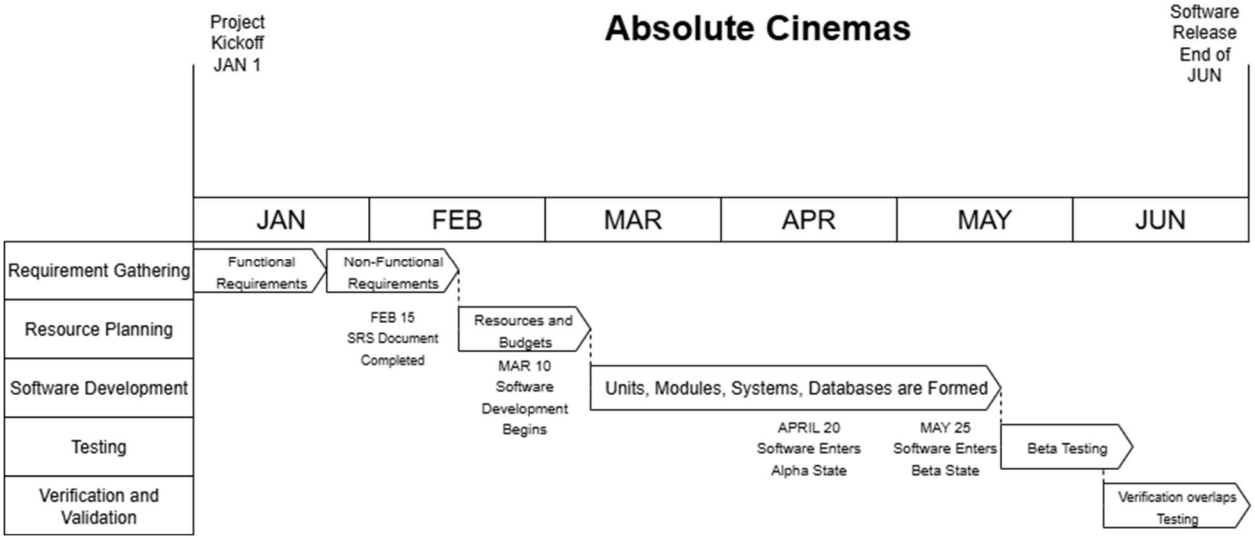


Figure 7

The development process of Absolute Cinemas is assumed to begin at the beginning of the year and is estimated to finish in the span of 6 months.

Requirement Gathering is the first stage and is complete when all Functional and Non-Functional Requirements are gathered in the SRS document. Resource Planning is designed to secure and establish available assets, including personnel, budgets, time constraints, and space. The Software Development stage is noticeably longer than the previous stages, as it involves the full development of the necessary features from the ground up. The testing phase brings together the hierarchy of developers to review every unit and module produced in the development stage. Finally, Verification and Validation will be executed by senior developers, engineers, and main clients to garner a satisfactory review of the final product as it nears the end of development.

5. Change Management Process

The SRS will be updated by the most senior developers with the proper digital key approved by upper management. Cases where the scope of the project shifts would involve a complete overhaul of the GitHub repository. Other cases that involve changes to requirements demand meticulous updates to each of the different sections for the requirements of the whole project through the GitHub repository. The approval process for requirement changes necessitates a minimum approval rate of 66.67% from upper management. Project shifts would warrant the approval of the entire upper management.

6. Testing Process

6.1 Introduction

The Testing Process ensures application reliability, error prevention, data protection, and improvements of user experience through rigorous software testing with numerous test cases, both big and small. Testing occurs in the timeline following the development phase of the

project. This means that Alpha testing will occur internally as modules become available, and Beta testing will be assigned as the main testing procedure. Along with third-party testing, junior and senior developers will take higher-priority test cases

6.2 Test Cases

The Test Case Template (6.1) is shown below. The table describes the process and the results of each test case.

FULL LIST OF TEST CASES

The associated test cases have been reviewed by senior developers at the highest priority level. The results of these test cases are beyond satisfactory, as seen in each status report. The highest priority tests ensure the security of the system and the integrity of the rest of the modules. Integration tests are assigned the next tier priority and provide the necessary data to couple modules and safely apply interdependencies. Finally, Unit tests are the lowest priority tests performed and reviewed by junior developers. Test vectors were implemented to ensure as much test coverage as possible during the given time.

6.3 Testing Procedures

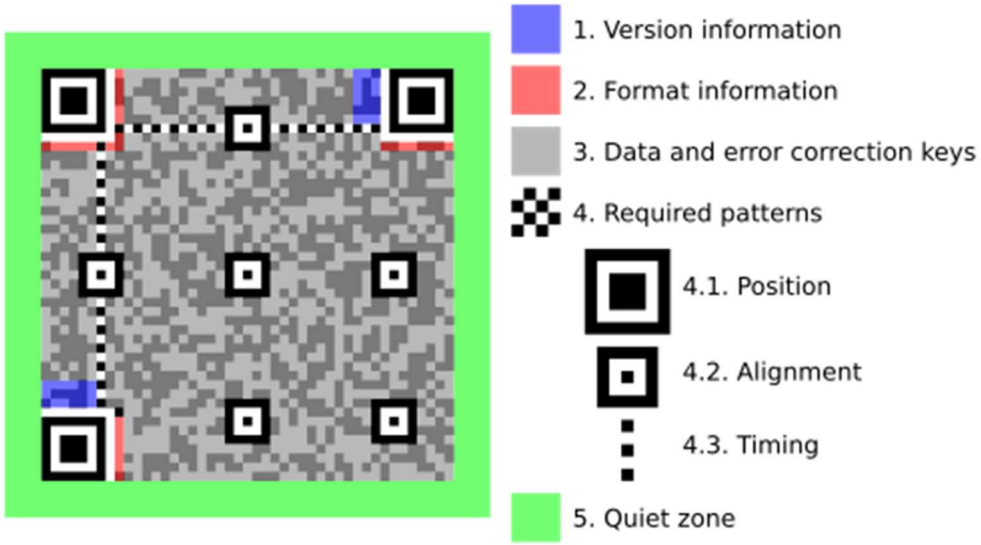
Testing Procedures explains the scope and effort of each test case provided in section 6.2 to a greater degree. Unit testing is typically performed automatically, with junior developers reviewing the results as well as developing new factors for testing. Integration testing has junior developers manually go through each of the variable cases to verify functionality of a specific function or functions. System testing is reserved for the most senior developers to test the integration of multiple functions and applications for distribution.

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

A.1 Appendix 1



Footnote: The structure of a version 7 QR code that highlights the essential functional elements.

A.2 Appendix 2

FAQ:

1. Are resold tickets from online sources that differ from the official mobile application or website accepted?

Unfortunately, we do not accept resold tickets; tickets will only be accepted through purchases from the official source.

2. Can I purchase food during the ticket purchase process?

Concessions can only be purchased through on-site movie theaters.

3. Can I get a refund for the ticket I have purchased?

The refund policy allows refunds 1 hour before the screening of the specific movie that the ticket is under.

4. Is my information being sold to third parties by using your services?

As mentioned in the User Privacy Policy (3.7 Design Constraints under Industry & Legal standards), we do not sell any information gathered from the user to third parties.