Software Requirements Specification

for

Video Rental System

Prepared by: ***Team Binary Brains***

Akshat Pandey (22CS10005)

Abhinav Akarsh (22CS30004)

Pranav Jha (22CS30061)

March 24, 2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

This document describes the Video Rental System(VRS). It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate, and how the system will react to external stimuli. This document is intended for the system's stakeholders and developers and will be proposed to the Software Engineering Laboratory for its approval.

1.2 Document Conventions

Bold text is used for software frameworks and tools. *Italicized* text is used for specific end-users of the application and classes.

1.3 Intended Audience and Reading Suggestions

Intended Audience:
Project stakeholders, development team, and documentation team involved in software development.
Reading Suggestions:
For stakeholders: "Understanding Software Requirements" for an overview.
For the development team: "SRS Guidelines" for writing precise requirements.
For the documentation team: "Technical Writing Essentials" for writing effective SRS documents.

1.4 Project Scope

The owner of a local movie store wanted to create a new business plan where everything about renting a *movie* (except picking up and returning the movie) was done online. Therefore, the new VRS will allow the following functionality online: to search for *movies*, to become members, to rent *movies*, to modify membership information, and to pay overdue fees. The store personnel may use the VRS to process the rented or returned *movies*, to add or remove *movies* to/from his store's *movies* inventory, and to update video information. The VRS is intended to increase the owner's profit margin by increasing *movie* sales with this unique business approach and by allowing him to reduce the staffing needed in his stores.

1.5 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998

2. Overall Description

2.1 Product Perspective

The video rental system connects us with our customers and helps us provide the *movies* and shows they love. It's not just about renting DVDs or streaming *movies*—it's about making sure everyone, from our customers to our employees, can easily find what they're looking for and enjoy it hassle-free. Our main goal is to keep our customers happy, make our operations smooth, and bring in more money by offering great entertainment options. This system allows customers to browse our collection, pick what they want, and start watching. Meanwhile, our *staff* can track what's available and help customers if needed. We know things change fast in technology and entertainment, so we're always ready to update our system to keep up with the latest trends and rules. Our video rental system ensures everyone gets the best entertainment experience possible, no matter how things change.

2.2 Product Features

- *User* Authentication
- Browse Movies
- Search Functionality
- Movie *cart* for customers
- Rental Management
- *User* Profiles
- *Staff* Dashboard
- Inventory Management
- Rental Tracking
- Reporting and Analytics
- Notifications

2.3 User Classes and Characteristics

The following *user* classes have been differentiated based on the subset of product features used. Each class poses a different interface in the application and is exposed to functions that are relevant to them.

- Customer: Browse and select *movies* for renting or purchasing, manage their shopping *cart*, view *order* history, and update account settings.
- Staff: Monitor and manage inventory levels, process *customer* orders and inquiries, provide support for technical issues, and assist with rental returns.
- Manager: Perform system audits, analyse sales and rental data, update *movie* listings and pricing, manage *staff* roles and permissions, and generate reports on system performance.

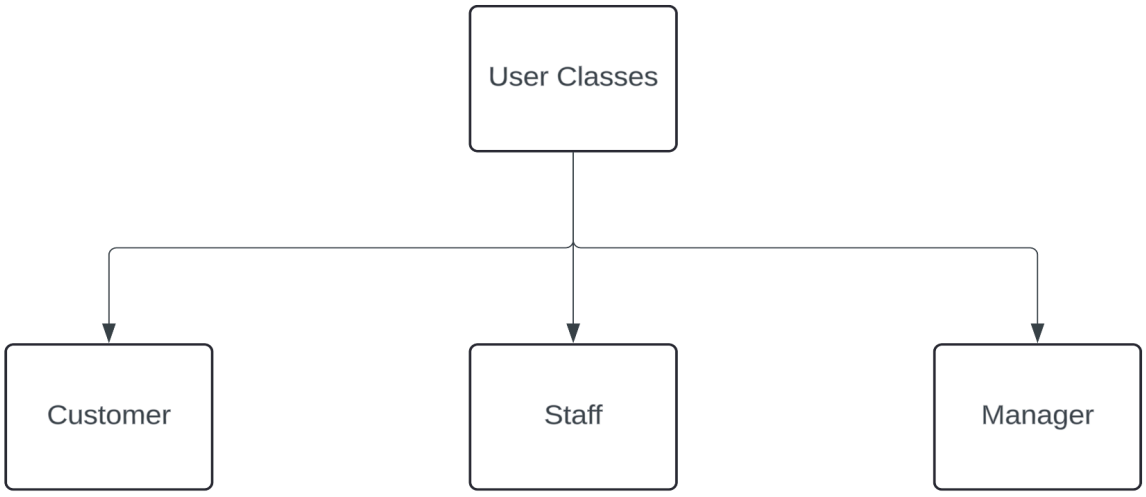

Fig: 2.1 User Classes

2.4 Operating Environment

The application is designed to use the platform-agnostic **Python** interpreter and the associated frameworks. The application server that handles incoming requests can be run on any platform with a TCP/IP stack supporting **Django** and **Python**. The web application can, therefore, be run on **Windows Server** or **Linux-based** distributions that support the above tools.

2.5 Design and Implementation Constraints

Developers are mandated to utilise **Python** for the backend services of the web application due to specific constraints:

- In adherence to professorial guidelines, developers are restricted to using **C++**, **Java**, or Python. This precludes the option of solely employing **HTML**, **CSS**, and **JavaScript** for frontend development, coupled with JavaScript for backend tasks (e.g., **Node.js**).
- **C++** and **Java** lack readily available, comprehensive frameworks akin to **Python's**. Given the project's tight deadline of less than two weeks, reliance on **Python's** pre-existing libraries and tools is imperative for swift development.

Considering these limitations, **Python** and the **Django framework** have been chosen as the most suitable solution for the web application.

2.6 User Documentation

The web application will possess a self-documenting, easy-to-use interface that does not require a specific set of manuals for any *user*. All the relevant information regarding web application usage will be placed at the appropriate locations inside the different web pages and displayed when needed.

2.7 Assumptions and Dependencies

2.7.1 Assumptions

- Internet Connectivity: It is assumed that the Video Rental System (VRS) *users* have a stable Internet connection to browse the *movie* catalogue, place orders, and access their accounts without interruptions.
- Data Security: It is assumed that the VRS employs robust security measures to protect *user* data, including encryption of sensitive information, secure payment processing, and protection against unauthorised access.
- Content Availability: The availability of *movie* titles within the VRS is subject to licensing agreements and partnerships with content providers. The system assumes the availability of a diverse range of *movies* for rental or purchase.
- Legal Compliance: The VRS assumes compliance with relevant laws and regulations governing online content distribution, copyright, and data privacy to ensure lawful operation.
- Dummy Payment Gateways: Integration with payment gateways (e.g., **Paytm, UPI**) is necessary to facilitate secure online payments for *movie* rentals or purchases within the VRS. We have assumed that secured payment is being taken care of.

2.7.2 Dependencies:

- **Python** and **Django** Framework: The backend development of the VRS relies on **Python** programming language and the **Django** framework for building web applications.
- Database Management System: The VRS depends on a compatible database management system, **SQLite3**, to store *movie* information, *user* data, and transaction records securely.
- Frontend Technologies: Dependency on frontend technologies such as **HTML, CSS,** and a bit of **JavaScript** for creating a user-friendly interface and interactive *user* experience.
- Content Providers: Dependency on partnerships with content providers and licensing agreements to access a diverse catalogue of *movies* for rental or purchase within the Video Rental System.
- Hosting Platform: The deployment and hosting of the VRS depend on a reliable hosting platform that supports **Python**-based applications and provides the necessary infrastructure for scalability and performance optimization.
- Development Tools: Dependency on development tools such as code editors, version control systems (e.g., **Git)**, and project management tools to facilitate collaborative development and streamline the development process of the VRS.

3. System Features

3.1 User Authentication

3.1.1 Description and Priority

- This feature involves the authentication process for *users* accessing the Video Rental System.
- It verifies the identity of *users* logging into the system. The message is redirected to the *staff* home page if the *user* is a *staff* member. Otherwise, it is redirected to the *customer's* home page.
- If the *user* is a *Manager*, it must log in to the admin portal.
- This feature is highly prioritised.

3.1.2 Stimulus/Response Sequences

- Stimulus: *User* enters login credentials.
  Response: The system validates the credentials.
- Stimulus: *User* attempts to access restricted features without logging in.
  Response: The system redirects the *user* to the log-in page.
- Stimulus: *User* enters incorrect credentials.
  Response: The system displays an error message indicating invalid login credentials.

3.1.3 Functional Requirements

- REQ-1: The system shall provide a login interface with fields for username and password.
- REQ-2: Upon submission of login credentials, the system shall authenticate the user.
- REQ-3: If the username or password is incorrect, the system shall display an error message and prompt the *user* to re-enter credentials.
- REQ-4: After successful authentication, the system shall grant access to the *user's* authorised features.
- REQ-5: The system shall enforce password complexity rules (e.g., minimum length, use of special characters) during *user* registration.
- REQ-6: The system shall implement secure encryption for storing *user* passwords.

3.2 Browse Movies

3.2.1 Description and Priority

- This feature enables *users* to browse the available *movies* in the Video Rental System.
- This feature is highly prioritised.

3.2.2 Stimulus/Response Sequences

- Stimulus: The *user* selects the "Genre" option from the main menu.
  Response: The system displays a dropdown menu of various *movie* genres.
- Stimulus: The *user* selects a genre to view *movies* of that genre.
  Response: The system displays *movies* of the selected genre.
- Stimulus: The *user* clicks on a *movie* for more details.
  Response: The system displays additional information about the selected *movie*, such as runtime, cast, director, release year, rating, certification, summary and price.
- Stimulus: The *user* enters the keywords to search for a *movie*.
  Response: The system displays the *movies* whose title, cast or genre matches the query keyword.

3.2.3 Functional Requirements

- REQ-7: The system shall present *users* with a searchable catalogue of available *movies*.
- REQ-8: *Users* can filter *movies* by genre and other relevant criteria.
- REQ-9: Each *movie* entry shall display essential details, including title, cover image, rating, cast, director and price.
- REQ-10: The system shall support infinite scrolling to accommodate large *movie* libraries.
- REQ-11: *Users* shall have the option to sort *movie* listings by release date or rating.
- REQ-12: Clicking on a *movie* shall display additional information, including summary, cast, director, runtime, and price.
- REQ-13: The system shall ensure that *movie* details and availability are updated in real time to reflect inventory changes.

3.3 Shopping Cart and Checkout

3.3.1 Description and Priority

- Users can add *movies* to a shopping *cart* and complete multiple rentals or purchases in a single transaction.
- This feature is highly prioritised.

3.3.2 Stimulus/Response Sequences

- Stimulus: The *user* selects a *movie* and clicks the "Add to Cart" button.
  Response: If the *movie* is available, the system will display a success message and add the selected *movie* to the user's shopping *cart*. Otherwise, the system shows an error message informing the *user* that the *movie* is out of stock.
- Stimulus: The *user* navigates to the shopping *cart* page to review selections.

Response: The system displays the list of *movies* in the shopping *cart* and the total cost. If a particular *movie* is out of stock, then the *movie* gets temporarily removed from the *cart* and added again to the *cart* once the *movie* is stocked up.

- Stimulus: The *user* removes items from the shopping *cart*.
  Response: The system recalculates the total cost based on the updated *cart* contents.
- Stimulus: *User* proceeds to checkout.
  Response: The system shows the *user* the list of *movies* in the *order* and prompts the *user* to enter payment details.
- Stimulus: *User* submits payment.
  Response: The system validates and processes the payment transaction.
- Stimulus: Payment transaction is successful.
  Response: The system generates an *order* confirmation, granting access to rented/purchased *movies*.
- Stimulus: *User* cancels the checkout process.
  Response: The system redirects the *user* to the shopping *cart* page without completing the transaction.
- Stimulus: *User* completes the checkout process.
  Response: The system updates inventory and *order* history and confirms successful transactions. Also, the user's *cart* becomes empty again.

3.3.3 Functional Requirements

- REQ-14: Implement shopping *cart* functionality with add/remove item options.
- REQ-15: Display *cart* summary with details of selected *movies* and total cost.
- REQ-16: Allow users to remove items from the *cart* before checkout.
- REQ-17: Provide a secure checkout process with payment options and *order* confirmation.

3.4 Manager Dashboard

3.4.1 Description and Priority

- *Managers* can access a dashboard to oversee various aspects of the Video Rental System (VRS), including inventory management, *user* details, *staff* details, *movie* details, and *order* details.
- This feature is highly prioritised.

3.4.2 Stimulus/Response Sequences

- Stimulus: The *manager* logs in to the *manager* dashboard.
  Response: The system authenticates *manager* credentials and grants access to managerial features.
- Stimulus: *Manager* views inventory status.

Response: The system displays current inventory levels, including available *movies* and their availability status.

- Stimulus: The *manager* accesses sales reports.
  Response: The system generates and presents sales reports, including revenue trends, popular *movie* titles, and *user* purchase patterns.
- Stimulus: The *manager* reviews *user* activity.
  Response: The system provides insights into *user* engagement metrics, such as registration rates, login frequency, and browsing behaviour.
- Stimulus: The *manager* performs inventory management tasks (e.g., adding new titles, updating availability).
  Response: The system facilitates inventory management actions and updates inventory records accordingly.
- Stimulus: The *manager* analyses *user* feedback and reviews.
  Response: The system aggregates and presents *user* feedback data for analysis, enabling *managers* to identify areas for improvement.

3.4.3 Functional Requirements

- REQ-18: The system must authenticate *manager* credentials securely before granting access to the *manager* dashboard.
- REQ-19: Provide real-time updates on inventory levels, displaying available *movies* and their availability status.
- REQ-22: Provide tools for *managers* to perform inventory management tasks, such as adding new titles, updating availability, and removing outdated content.

3.5 Staff Inventory Management

3.5.1 Description and Priority

- *Staff* members are equipped with tools to manage the inventory of *movies*, including stocking up the *movies*.
- This feature is highly prioritised.

3.5.2 Stimulus/Response Sequences

- Stimulus: *Staff* member logs in to the *staff* dashboard.
  Response: The system authenticates *staff* credentials and grants access to inventory management tools.
- Stimulus: *Staff* member navigates to the *Movies* section.
  Response: The system lists all *movies,* available and rented quantities, and inventory management options.
- Stimulus: *Staff* member selects a *movie* title to increase or decrease its quantity.
  Response: The system updates the stock of the specified *movies*.
- Stimulus: *Staff* members select the “Orders” option and a sub-option for ‘all’, ‘rented’, and ‘bought’.

Response: The system displays a list of current orders of a given type containing *user*, *movie*, *order* date, due date and current status.

3.5.3 Functional Requirements

- REQ-24: Authenticate *staff* credentials securely before granting access to inventory management tools.
- REQ-25: Provide *staff* with an option to increase and decrease stock quantity.
- REQ-26: Allow *staff* to update the current status of an *order* if the customer returns a *movie*.

3.6 User Profile

3.6.1 Description and Priority

- The *user* profile feature allows registered *users* to create and manage their accounts within the Video Rental System (VRS).
- *Users* can customise their profiles and view their previous orders.
- This feature has medium priority.

3.6.2 Stimulus/Response Sequences

- Stimulus: *User* navigates to the profile settings.
  Response: The system displays options for profile customization and updating personal information.
- Stimulus: *User* views their *order* history.
  Response: The system lists past rental activities, including rented and bought *movies*.

3.6.3 Functional Requirements

- REQ-27: Profile Customization allows *users* to customise their profiles by updating personal information.
- REQ-28: Display a history of the *user's* past *order* details, including rented and bought *movies*.

3.7 Notification System

3.7.1 Description and Priority

- The notification system feature enables the Video Rental System (VRS) to communicate important information, updates, and alerts to users promptly and effectively.
- Notifications can include reminders for upcoming rental expirations to the *customers*.

- *Staff* is notified about the *movies* currently absent from the inventory.
- This feature has medium priority.

3.7.2 Stimulus/Response Sequences

- Stimulus: The system detects an upcoming rental expiration.
  Response: The system sends a notification to the *user*, reminding them of the impending rental expiration and prompting action to return the rented *movie*.
- Stimulus: Some *movies* become out of stock.
  Response: The system sends *staff* notifications about the *movies* that are currently out of stock in the inventory.

3.7.3 Functional Requirements

- REQ-29: Implement mechanisms to trigger notifications when the rental expiration date is near.
- REQ-30: Implement mechanisms to notify *staff* when some *movie* is out of stock.

3.8 Recommendation System

3.8.1 Description and Priority

- The recommendation system feature enhances the *user* experience within the Video Rental System (VRS) by providing personalised movie recommendations based on search history, purchase history and current purchase trends.
- By leveraging data analytics, the recommendation system suggests relevant and engaging movies to *users*, encouraging exploration and increasing *user* satisfaction.
- This feature has low priority.

3.8.2 Stimulus/Response Sequences

- Stimulus: *User* logs in to their account.
  Response: The system analyses *user* profiles and viewing history to generate personalised movie recommendations for the *user*.
- Stimulus: *User* browses movie catalogues or search results.
  Response: The system displays recommended movie titles alongside search results or within the movie catalogue, tailored to the *user's* preferences and interests.
- Stimulus: *User* adds to cart, rents or purchases a recommended movie.
  Response: The system tracks *user* interactions and adjusts recommendations based on viewing behaviour, promoting similar or related movie titles for future recommendations.
- Stimulus: The *user's* feed gets updated according to the recommendations.
  Response: The system searches for the best matching movies for the *user* and updates the *user's* feed accordingly.

3.8.3 Functional Requirements

- REQ-31: Collect and analyse *user* data, including viewing history, ratings, reviews, and preferences, to build comprehensive *user* profiles for personalised recommendations.
- REQ-32: Ensure diversity and serendipity in recommendations by incorporating a variety of movie genres, themes, and styles, encouraging *users* to discover new and unexpected titles.

4. External Interface Requirements

4.1 User Interfaces

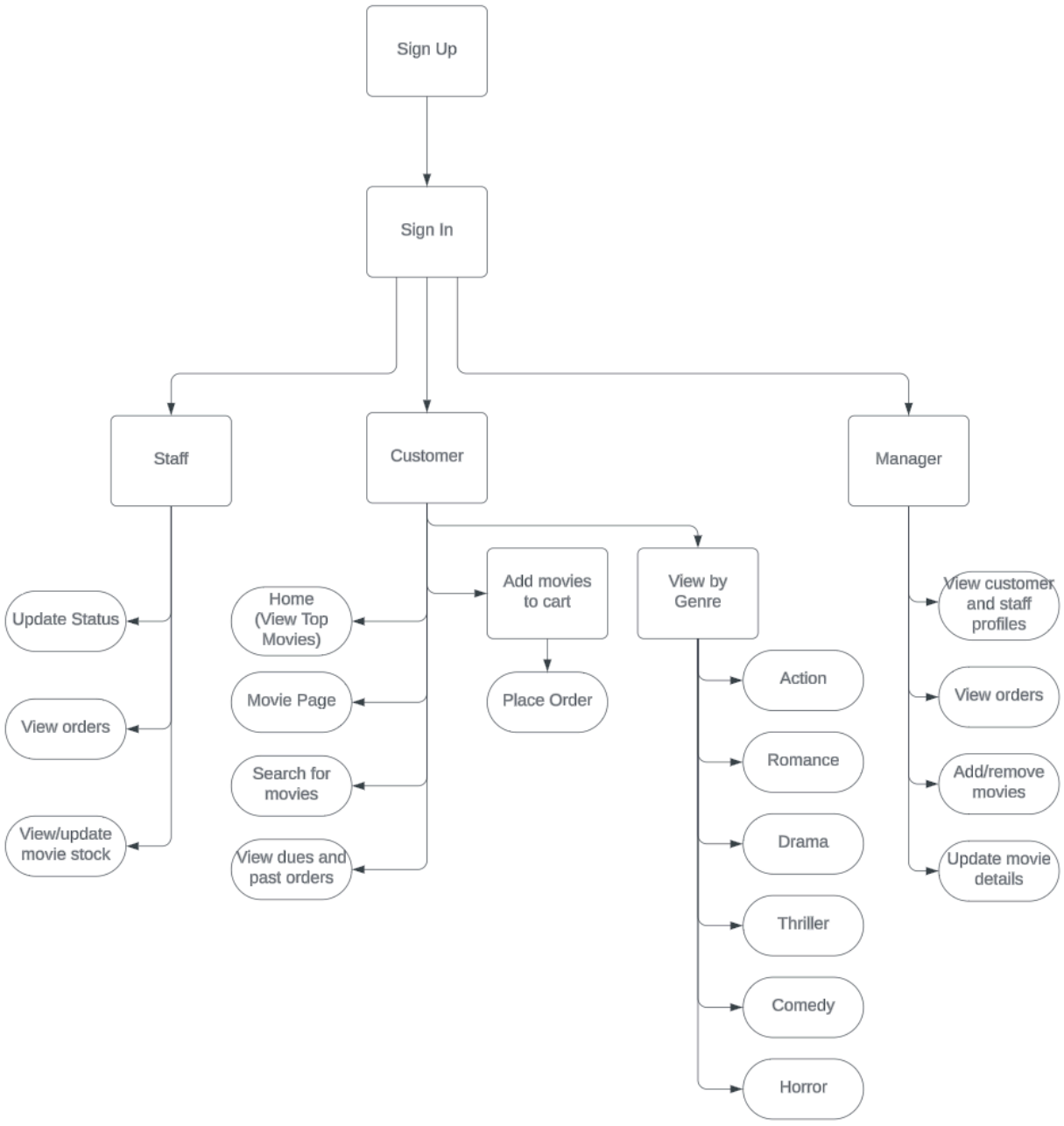

Fig: 4.1 User Interface Diagram

4.2 Hardware Interfaces

- This hardware must be a server computer with a fast processor to handle many requests simultaneously. A multicore system with distributed processes across the core would be preferable.
- Solid state drives instead of hard disks to boost the speed of information retrieval.
- Enables faster request handling.

4.3 Software Interfaces

- Web Browser: *Users* interact with the VRS front end using web browsers like **Google Chrome, Mozilla Firefox,** or **Safari.**
- Backend Framework **(Django**): The VRS backend is built using the **Django** framework, providing the necessary tools and libraries for developing web applications.
- Database Management System (**SQLite3**): The VRS interfaces with a database management system (DBMS) like **SQLite3** for storing and retrieving movie data, *user* information, and transaction records.

4.4 Communications Interfaces

- A standard TCP/IP stack will be used on most operating systems.
- Will use the HTTPS protocol to deliver webpages and handle *user* login.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Response Time

The system should respond well to *user* actions like browsing movies, renting, checkout, etc. The system should be able to handle searches and database queries efficiently.

5.1.2 Scalability:

The system architecture should be scalable to accommodate increasing numbers of *users* and movie inventory.

5.1.3 Data Retrieval Time:

Movie details, *user* profiles, cart items, *order* history, and due dates should be retrieved from the database as fast as possible.

5.1.4 Concurrency Handling:

The system should support concurrent access by multiple *users* without data integrity issues. Concurrent rental transactions should be handled gracefully without conflicts or errors.

5.2 Safety Requirements

There needs to be a regular audit since there is no check for the stock information the *staff* enters. Moreover, an audit also needs to be done for the sold and rented movies, as there might be cases of fraud in the long run if left unchecked since the system is not verifying the data.

5.3 Security Requirements

5.3.1 Secure Login

The system should be protected against forced and malicious login attempts to access the software. The web interface must be continuously patched to fix bugs as and when they crop up.

5.3.2 Verifiable Access

The system is designed to serve *customers* and the management team. The levels of access and their permissions are decided by their login IDs. Thus, the login and access to

functionality should be robust and highly secure to ensure the smooth operation of the business.

5.4 Software Quality Attributes

5.4.1 Reliability

The system must be dependable, which means it must constantly perform as expected with very low downtime and rate of failure. Without this, the VRS will not be able to function smoothly.

5.4.2 Information Accessibility

To save login credentials and the various types of data that the software uses, the application should be able to save, access, and alter files in the system to retain information even on successive runs.

5.4.3 Maintainability

The program should be easy to modify and add features to. The code should be commented on, and the features should be well-documented to allow easy maintenance. The frameworks used should be well supported by their developers well into the future.

Appendix A: Glossary

Video Rental System (VRS)	An online web platform allowing *users* to rent or purchase movies, manage inventory, and facilitate transactions
Customer	A *user* of the VRS platform who can browse, rent, and purchase movies.
Staff	Employees responsible for managing inventory, processing orders, and ensuring *customer* satisfaction within the VRS.
Manager	A *staff* member with administrative privileges overseeing inventory, transactions, and system operations.
Login	Accessing the VRS platform using credentials such as email ID and password.
Genres	Categories into which movies are grouped based on common themes or styles, facilitating browsing and search.
Movie Details	Information about movies, including genres, cast, crew, and plot summaries, is provided to *users* for informed decision-making.
Rent	Refers to temporary access to a movie for a specified duration in exchange for payment.
Buy	This involves permanently acquiring a movie for unlimited viewing, typically at a higher cost.
Cart	A virtual shopping cart where *customers* can add movies before purchasing or renting.
Order	A transaction initiated by a *customer* that contains details about the *user* who purchased/rented a movie and the total price.
Invoice	A document generated by the VRS system detailing the cost of rented or purchased movies is provided to *customers* for payment confirmation.
Inventory Management	*Staff* oversees the maintenance and replenishment of movie stock within the VRS to ensure availability and prevent stockouts.
Notifications	Messages the VRS system sends to *staff* regarding inventory status, rental durations, and other essential updates.

Customer Profiles	*User* accounts containing order history, rental due dates, and personal preferences are accessible for reference and customization.
Rental Duration	The maximum period for which the *customer* can keep a rented movie.
Inventory Stocking	The practice of replenishing movie stock to meet *customer* demand and prevent shortages is managed by *staff* to ensure availability.