Software Requirements Specification

for

OUTFIT PLANNER GAME

Version 1.0 approved

Prepared by Deshanae M, Jada J, Ludy J, Eunice A.

Farmingdale State College

10/07/2025

Table of Contents

Table of Contents ii
Revision History ii
1. Introduction 1
1.1 Purpose 1
1.2 Document Conventions 1
1.3 Intended Audience and Reading Suggestions 1
1.4 Product Scope 1
1.5 References 1
2. Overall Description 2
2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Classes and Characteristics 2
2.4 Operating Environment 3
2.5 Design and Implementation Constraints 3
2.6 User Documentation 3
2.7 Assumptions and Dependencies 4
3. External Interface Requirements 4
3.1 User Interfaces 4
3.2 Hardware Interfaces 5
3.3 Software Interfaces 5
3.4 Communications Interfaces 6
4. System Features 6
4.1 System Feature 1 7
4.2 System Feature 2 (and so on) 7
5. Other Nonfunctional Requirements 9
5.1 Performance Requirements 9
5.2 Safety Requirements 9
5.3 Security Requirements 10
5.4 Software Quality Attributes 10
5.5 Business Rules 10
6. Other Requirements 11
Appendix A: Glossary 11
Appendix B: Analysis Models 12
Appendix C: To Be Determined List 12

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of Stylz, a desktop dress-up game designed to let users express their creativity through avatar customization. It outlines the objectives, main features, and design requirements of the software.

1.2 Document Conventions

This SRS follows the IEEE 830-1998 standard for software requirements documentation. The document uses *Times New Roman* font, 12-point for body text and 14-point bold for section headings. Bold text highlights key terms and section titles; *italics* indicate notes or examples, and monospace font is used for code, file names, and database fields. Each requirement is assigned a unique identifier (e.g., *FR-1*, *NFR-1*) for traceability. Priorities are not inherited; every requirement specifies its own priority level as High, Medium, or Low.

1.3 Intended Audience and Reading Suggestions

This document is intended for individuals who are interested in using or learning more about Stylz. The primary users for Stylz are teens, students, or anyone who enjoys fashion and wants to express their creativity through this interactive dress-up game. It may also be useful for instructors or developers reviewing or expanding the project.

The remainder of the SRS describes the system's overall purpose, major features, interfaces, and performance requirements. Readers should begin with the introduction and overall description for context, then move to the system features and requirements sections for more detailed information.

1.4 Product Scope

Stylz is a desktop dress-up game that allows users to customize an avatar using a wide range of clothing and accessory options. The goal of the game is to provide a fun and creative outlet where users can experiment with different styles and combinations.

1.5 References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Stylz's GitHub page:
https://github.com/DeshanaeMorris/Outfit-Planner-Project.git

Figma:
https://www.figma.com/design/rrt094ykF7UY9T2isRI8jG/Untitled?node-id=0-1&p=f&t=vDSuIZv3t2Y9Wbjs-0

IEEE Template for System
https://goo.gl/nsUFwy

2. Overall Description

2.1 Product Perspective

Stylz is a web application designed for users who enjoy creating and exploring fashion designs with avatars. It allows users to design and customize outfits with a variety of clothing and accessory options. This outfit planner is a standalone product, developed from the ground up and designed to run directly from any web browser.

Some of the game's features include account creation, avatar customization, and the ability to save and rate outfits. It is developed with JavaFX, HTML, and database integration to store user information and outfit data. To create clothing and accessory illustrations, we will use Picsart, Figma, Canva, or any other source of inspiration.

2.2 Product Functions

Major functions of Stylz game:

- **User Account Management:**
    - Allow users to Sign up, log in, and secure user data.
- **Avatar Customization:**
    - Users can customize their own avatar using various clothing options such as tops, bottoms, shoes, and accessories.
- **Outfit saving:**
    - A database that allows users to save their designs and view them later in their personal gallery.
- **Outfit Rating:**
    - Users can rate and review their saved outfits.
- **Random Outfits:**
    - The system will generate random outfit combinations to inspire users.
- **Reset Function:**
    - The system will allow users to reset their avatar to start a new design from scratch.
- **Database Storage:**
    - All user information, outfits, and rating data will be stored in the database.

2.3 User Classes and Characteristics

- **General Users:**
    - This game can be played by any person who wants to create and explore outfit ideas.
- **Designers:**
    - Developers or artists who create new clothing and accessories assets for the game to keep the game updated.

2.4 Operating Environment

- ★ **This is a web-based game and will operate in any Operating System.**
    - Windows, macOS, Linux?
- ★ **Browser:**
    - It can be accessed through any web browser such as Google Chrome, Firefox, Edge, or Safari.
- ★ **Tools Used**:
    - JavaFX
    - HTML
- ★ **Database:**
    - MySQL
- ★ **Design Tools:**
    - Picsart
    - Figma
    - Canva
    - Internet sources-Pinterest outfit inspiration.
- ★ **Hardware Requirements:**
    - Any computer or laptop with internet connection.

2.5 Design and Implementation Constraints

The development of Stylz is subject to have several constraints that limit designs and styling. The project shall be implemented using JavaFX and HTML for the interface and MySQL for backend data management. To develop this project, we are using SceneBuilder for GUI and IntelliJ IDEA for java development. All user data shall be stored securely in the MySQL database to protect personal information and save designs. To design, we are using our own content or other sources. Users aren't allowed to upload their own content due to security reasons.

2.6 User Documentation

The application will include several components:

- **User Manual (PDF):**
    - The manual is a comprehensive step-by-step guide that explains how to install, launch, and use Stylz. This guide also includes instructions for creating an account, customizing avatars, saving, generating, and rating outfits.

- **Help:**
    - The system will have an integrated tab with frequently asked questions and brief explanations of each feature.

- **Developer Documentation:**
  - Technical documentation will be stored in the GitHub repository. This will include the code, organization, structure, and design of the application for future updates.

- **Video Tutorial:**
  - The application will display a tutorial to visually guide new users through basic Stylz functions, such as logging in, navigating menus, providing outfit examples, and more.

2.7 Assumptions and Dependencies

Stylz depends on several assumptions and external factors that may influence system performance and maintenance.

- **System Requirements:**
  - It is assumed that users will have access to a stable internet connection and a device capable of running JavaFX-based applications through modern browsers such as Chrome, Firefox, or Edge.

- **Database Connectivity:**
  - The MySQL database must remain active and properly configured to store and retrieve user accounts, avatar designs, and ratings.

- **Third-Party Tools:**
  - The project uses software such as SceneBuilder, IntelliJ IDEA, Figma, Canva, and Picsart for development and design purposes.

3. External Interface Requirements

3.1 User Interfaces

 The user interface will be developed using **JavaFX** and designed in **SceneBuilder**.

 The interface will consist of multiple scenes:

Home Screen – Displays app title, “Start” button, catalog, and brief instructions.

Login/Sign up Screen – Allow users to create or login to an account that saves their progress

Shop Screen – Displays a list or grid of clothing items with images, price, and “checkout” button. The user’s virtual budget is displayed at the top.

My Closet Screen – Shows items purchased by the user and allows them to drag and drop items.

Users can **drag and drop** clothing images onto the mannequin area to visualize an outfit.

A “Save Outfit” button records the outfit details.

Catalog Screen – Displays all previously saved outfits in a list format.

The app will use consistent button styles (rounded edges, hover color change) and a light color theme.

Standard GUI elements include:

Buttons: “Start,” “Buy,” “Create Outfit,” “Save,” “Back.”

Labels and text fields for item names, prices, and totals.

Message pop-ups for success or budget-limit warnings.

Error messages will appear as JavaFX alerts (e.g., “Insufficient budget,” “No item selected”).

3.2 Hardware Interfaces

The system will primarily run on standard desktop or laptop computers capable of running the Java Runtime Environment (JRE).

- **Logical Characteristics:**
  The application will not directly interact with external hardware beyond standard input/output devices such as the mouse, keyboard, touchscreen, and display. User actions like clicking, scrolling, and drag-and-drop will be captured through browser-based events, while the interface will render visually through standard HTML, CSS, and JavaScript on the client side.
- **Physical Characteristics:**
  Minimum requirements include a dual-core processor, 4 GB of RAM, and 200 MB of available storage space. The application will require a screen resolution of at least 1280×720 for optimal layout display and a stable internet connection.
- **Data and Control Interactions:**
  All user inputs are processed through the browser and communicated to the server via standard HTTP requests.
- **3.3 Software Interfaces**

The system interacts with several software components to provide full functionality:

- **Operating System:**
  The app is compatible with Windows, macOS, and Linux platforms that support Java 17 or later.
- **Programming Environment:**
  Developed in **Java** using **IntelliJ IDEA** as the IDE. The GUI is implemented with **JavaFX** and designed using **SceneBuilder**.

- **Database Interface:**
  The application connects to an **SQLite** database to store user information, saved outfits, and purchase history. SQL queries will handle data retrieval and updates. Data such as usernames, clothing IDs, image paths, and outfit combinations are passed between the Java backend and the database using JDBC.
- **Libraries and Tools:**
  - JavaFX (for GUI rendering)
  - JDBC (for database communication)
- **Data Flow:**
  Input data (user credentials, clothing selections) are sent from the user interface to the backend logic. Output data (saved outfits, updated budget) are retrieved from the database and displayed in the GUI.
- **Services and Communication:**
  Internal communication between modules follows a Model-View-Controller (MVC) structure. The Model manages data logic, the View handles display, and the Controller processes user input and updates the Model.

3.3 Communications Interfaces

The application will require an internet connection to function since it is a web-based system.

- **Network Communication:**
  The user's web browser connects to the server through HTTP or HTTPS. The browser sends requests for data such as login information, clothing items, or saved outfits, and the server sends responses back.
- **Communication Standards:**
  The app will allow data to be exchanged through simple web requests.
  - Requests are used to load data (like items or saved outfits).
  - Requests are used to send data (like creating an account or saving an outfit).
- **Message Formatting:**
  The information sent between the front end and back end will be formatted so it's easy for both the browser and server to read.
- **Security:**
  The system will use **HTTPS** to keep data private and secure. User passwords and other sensitive information will be protected during transmission.
- **Data Updates:**
  When a user performs an action, such as saving an outfit or buying an item, the browser updates the database through the server, so data stays current.

4. System Features

4.1 System Feature 1

4.1.1 Description and Priority

This feature allows the user to select clothing items (e.g., tops, bottoms, shoes, accessories) from a library of available outfits.
Priority: High

4.1.2 Stimulus/Response Sequences

- **User Action:** User opens the outfit selection screen.
  System Response: The system displays categories of clothing items.
- **User Action:** User selects a category (e.g., Tops).
  System Response: The system displays available tops.
- **User Action:** User clicks on a clothing item.
  System Response: The item is added to the outfit preview.

4.1.3 Functional Requirements

- **REQ-1:** The system shall display clothing items organized by categories (tops, bottoms, shoes, accessories).
- **REQ-2:** The system shall allow the user to preview a selected item on the character model.
- **REQ-3:** The system shall prevent multiple items of the same category from being applied simultaneously (e.g., only one pair of shoes).
- **REQ-4:** The system shall display an error message if a user attempts to select an invalid or unavailable item.

4.2 System Feature 2 (and so on)

4.2.1 Description and Priority

Users can save their created outfits to view or load later.
Priority: High

4.2.2 Stimulus/Response Sequences

- **User Action:** User clicks “Save Outfit.”
  System Response: The system prompts for an outfit name and saves it.
- **User Action:** User clicks “Load Outfit.”
  System Response: The system displays a list of saved outfits to choose from.

4.2.3 Functional Requirements

- **REQ-5:** The system shall allow users to save an outfit with a unique name.
- **REQ-6:** The system shall provide a list of saved outfits for loading.
- **REQ-7:** The system shall overwrite an outfit if the user saves with an existing name (after confirmation).
- **REQ-8:** The system shall display an error message if the user tries to save without naming the outfit.

4.3 Random Outfit Generator

4.3.1 Description and Priority

The system generates a random outfit combination from the available clothing items.
Priority: Medium

4.3.2 Stimulus/Response Sequences

- **User Action:** User clicks “Randomize Outfit.”
  System Response: The system selects one random item from each category and displays the outfit.

4.3.3 Functional Requirements

- **REQ-9:** The system shall randomly generate an outfit by choosing one item from each category.
- **REQ-10:** The system shall ensure that only valid clothing combinations are applied.
- **REQ-11:** The system shall display a new random outfit each time the feature is used.

4.4.1 Description and Priority

Users can customize the base character (e.g., skin tone, hair, body type).
Priority: Medium

4.4.2 Stimulus/Response Sequences

- **User Action:** User clicks “Customize Character.”
  System Response: The system displays options for appearance customization.

- **User Action:** User selects hair color.
  System Response: The system updates the character preview in real time.

4.4.3 Functional Requirements

- **REQ-12:** The system shall provide customization options for skin tone, hair, and body type.
- **REQ-13:** The system shall update the character preview immediately when changes are applied.
- **REQ-14:** The system shall reset to default customization if requested by the user.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **PR-1:** The system shall load all core assets (character model, clothing categories, and menu interface) within 5 seconds of startup.
- **PR-2:** The system shall display selected clothing items in the outfit preview within 1 second of user interaction.
- **PR-3:** The system shall support at least 100 clothing items in the library without noticeable lag in rendering or navigation.
- **PR-4:** The system shall maintain a frame rate of at least 30 FPS on standard desktop hardware.
- **PR-5:** The system shall use asynchronous loading for large image files to prevent freezing during gameplay.

5.2 Safety Requirements

- **SR-1:** The system shall prevent file overwriting or corruption when users save or update outfit data.
- **SR-2:** The system shall validate all external API data before use to avoid crashes or unexpected behavior.
- **SR-3:** The system shall ensure that the user cannot delete or modify core system files (e.g., default clothing assets).
- **SR-4:** The system shall display warnings before performing actions that might cause data loss (e.g., clearing saved outfits).
  Rationale: Even though this is a low-risk application, basic data safety and integrity must be maintained to protect user creations and ensure system stability.

5.3 Security Requirements

- **SEC-1:** The system shall restrict file access so that only authorized users can modify the asset or data directories.
- **SEC-2:** If connected to an online API, the system shall use secure HTTPS requests to protect data transmission.
- **SEC-3:** The system shall not collect or store personal information from users.
- **SEC-4:** The system shall sanitize all API responses to prevent injection of malicious code or invalid data.
  Rationale: Maintaining user trust and protecting data integrity are essential, especially if external connections or shared data are involved.

5.4 Software Quality Attributes

- **Adaptability:** The system shall be designed so that new clothing items, categories, or themes can be added without altering core code logic.
- **Maintainability:** The MVC architecture ensures that changes in the user interface or logic can be made independently.
- **Usability:** The user interface shall be intuitive, using labeled buttons and icons to guide players through outfit creation.
- **Portability:** The application shall run on Windows, macOS, and Linux systems with JavaFX 21 or later installed.
- **Reliability:** The system shall handle missing or corrupted image files gracefully, displaying fallback placeholders instead of crashing.
- **Testability:** The system shall allow individual component testing for models, controllers, and views to verify correct behavior.
- **Reusability:** The clothing asset loader and preview renderer shall be modular for reuse in future updates or related projects.

5.5 Business Rules

- **BR-1:** Only the system administrator or developer shall be allowed to add or remove clothing items from the library.
- **BR-2:** Users shall be allowed to save and load outfits locally but cannot modify system files directly.
- **BR-3:** Saved outfits must have unique names; duplicate names shall prompt an overwrite confirmation.
- **BR-4:** The system shall automatically refresh the clothing library each time the application starts.
- **BR-5:** When an API connection is used, only approved and verified data sources shall be accepted.

6. Other Requirements

Database Requirements:

- The system will use a **MySQL** database to store user accounts, outfit details, ratings, and customization data.
- All tables shall include primary keys, foreign keys, and constraints to maintain data integrity.
- Backup and recovery mechanisms shall be implemented to prevent data loss.
- Database queries must be optimized to ensure that read and write operations occur within **2 seconds** under normal load.

Internationalization Requirements:

- The initial release will support **English only**, but the system shall be designed to allow for easy integration of multiple languages in future updates.
- Text elements (e.g., button labels and messages) shall be stored in a configurable resource file to support translation.

Future Enhancements:

- Planned improvements include adding multiplayer fashion challenges, user profile pages, and integration with online fashion trend APIs.
- The system design shall remain flexible to accommodate these features without major restructuring.

Appendix A: Glossary

Term	Definition
Avatar	A digital character that represents the user within the game.
GUI (Graphical User Interface)	The visual part of the application that users interact with.
JavaFX	A Java library used to build rich graphical user interfaces.
MVC	Model-View-Controller design pattern that separates application logic, UI, and data handling.
MySQL	A relational database management system used to store and manage game data.
Outfit	A combination of clothing and accessories designed by the user.
Randomizer	A function that automatically generates a random outfit combination.

Appendix B: Analysis Models

★ **Data Flow Diagram (DFD):**
The DFD will illustrate how data moves between user inputs, the game interface, and the database.

★ **Entity-Relationship Diagram (ERD):**
The ERD will include entities such as *User*, *Outfit*, *Clothing Item*, and *Rating*, with relationships defined by user activity and data storage.

★ **Class Diagram:**
A UML class diagram to present core components, including *Avatar*, *Outfit Manager*, *Database Handler*, and *UI Controller*.

Appendix C: To Be Determined List

TBD ID	Item	Description	Status / Notes
TBD-1	Final UI color scheme	Decide the main color palette and theme for the interface	Pending design approval in Figma
TBD-2	Additional clothing categories	Determine if outerwear, hats, or seasonal items will be added	Planned for future release
TBD-3	API integration	Decide if online outfit inspiration API will be connected	Pending research
TBD-4	Multi-language support	Determine which additional languages will be supported	Future enhancement
TBD-5	Random outfit algorithm	Finalize the logic for generating random outfit combinations	Needs testing and verification
TBD-6	Asset licensing	Confirm all images, icons, and fonts are properly licensed	Pending legal check
TBD-7	Performance benchmarks	Set exact loading time and FPS requirements for various hardware	Pending performance testing