|  |
| --- |
| Grand Canyon University: Programming in Java III |
| Milestone 1 – Project Proposal, Sitemap, and Division of Work |
| Tasks Completed in Milestone 1: Selected project domain: Video Game Library Catalog Defined core entity: VideoGame (title, year, genre, description, award) Wrote detailed Project Proposal with domain and features Created Draft Sitemap Diagram showing pages and navigation Wrote Division of Work (Solo Plan) with Agile-lite approach Documented Technical & Functional Risks Added General Technical Approach and Key Technical Design Decisions |

|  |
| --- |
| Carlos Cortes  8-23-2025 |

**CST-339 Programming in Java III**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Milestone 1 – Project Proposal, Sitemap, and Division of Work | |
| **Date:** | 08/22/2025 | |
| **Revision:** | 1.0 | |
| **Team:** | 1. Carlos Cortes | |
|  | |
|  | |
|  | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | Milestone 1 Proposal, Sitemap, Division of Work | Carlos Cortes | 10 | 70+ | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | https://github.com/CarlosECortes/CST-339-Programming-in-Java-III | |
| **Screencast URL:** | *The URL that can be used to access your screencast demonstration video for the respective assignment.* | |
| **Peer Review:** | *NA* | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Planning Documentation**

**Initial Planning:**

My project is a Video Game Library Catalog that tracks “Game of the Year” titles from the last 20 years across three popular genres: Role Playing Games (RPG), First Person Shooter (FPS), and Action/Adventure. It will manage VideoGame entities with the following properties: id (auto-generated), title, genre, year, description.

**High-Level Features:** The users can register and log in. A Game List page will show all video games in a table. Users can add a new game, see details about a game, edit a game, and delete a game. Forms will check for errors (for example, the year must be a number). Pages will be built with Thymleaf and styled with Bootstrap so they look clean and work on any device. The games will be stored in a MYSQL database, and we will use Spring Data JDBC to connect to it.

**Retrospective Results:**

*The following table should be completed after each Retrospective on things that went well (keep doing).*

|  |
| --- |
| **What Went Well** |
|  |
|  |
|  |

*The following table should be completed after each Retrospective on things that didn’t go well (stop doing) and what would be done differently next time with an action plan to improve (try doing and continuous improvement).*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
|  |  |  |
|  |  |  |
|  |  |  |

**Design Documentation**

**General Technical Approach:**

We will build the project with Spring Boot. The application will have simple web pages made with Spring MVC, Thymeleaf, and Bootstrap for a clean look. Small service classes that handle the game catalog, like adding or updating a game. A MySQL database that stores all the games. We will use Spring Data JDBC to connect and run queries. Lastly, users must log in to see and change games. We will add Spring Security for login protection.

**Key Technical Design Decisions:**

We are choosing to use a Spring Boot app with an N-Layer structure (Pages, Services, Data) to meet the course requirements and keep UI, logic, and database separate for easier testing and changes. For the UI stack, we are using Spring MVC with Thymeleaf templates and Bootstrap for responsive design because the project requires it, and Thymeleaf keeps HTML simple and Bootstrap keeps pages mobile-friendly. For validation and errors, we will use basic server-side validation so that required fields like title and year are checked. An error message will display on the page.

**Risks:**

Learning Spring Security might be hard, so I will follow tutorials and examples step by step. Connecting to MySQL could cause errors. We will test a small sample database early. Because I am doing this individually instead of in a group, the workload could be heavy. To manage this, I will keep the scope limited to one entity (VideoGame) and keep features simple.

**Division of Work (Solo Approach):**

Since this is an individual project, all development tasks will be completed by me. To manage the workload, I will organize the project into layers and milestones as follows:

Front-End (Presentation Layer):

I will design Thymeleaf pages for login, registration, and CRUD (list, details, add, edit, delete). Pages will be styled with Bootstrap to ensure a clean, responsive look.

Back-End (Business Layer):

I will create the service classes that handle the main game catalog logic, such as adding, updating, deleting, and viewing games.

Database (Persistence Layer):

I will configure a MySQL database and set up Spring Data JDBC repositories to connect and run queries.

Security (Security Layer):

I will integrate Spring Security later in the project to ensure that only registered and logged-in users can manage games.

Shared Tasks (throughout milestones):

Testing, debugging, documenting design decisions, and managing risks will be done continuously.

**Sitemap Diagram:**

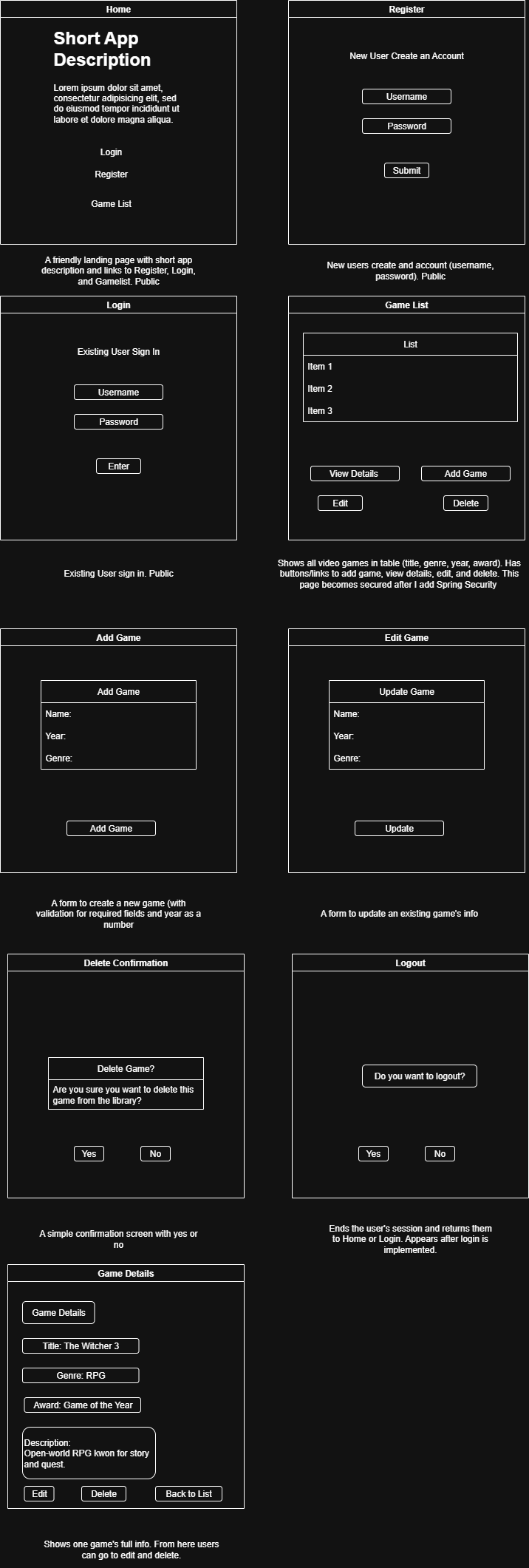
*.*

Figure 1. The sitemap shows how users move between public pages (Home, Register, Login) and secured pages (Games List, Details, Add, Edit, Delete).

**User Interface Diagrams:**

*You should insert any wireframe drawings or whiteboard concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for why you are able to leave this section as N/A.*

**Class Diagrams:**

*You should insert any class diagrams here. Your class diagrams should be drawn correctly with the 3 appropriate class compartments, + and – minus to indicate accessibility, and also the data types for the state/properties as well as method arguments and return types. If you have no supporting documentation, please explain the rationale for why you are able to leave this section as N/A.*

**Service API Design:**

*This section should fully document any service API’s (like REST API’s) that are being published, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by a third-party developer to integrate with the service and API. The design can also be captured with tools such as Swagger.*

**Security Design:**

*This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.*

**Other Documentation:**

*You should insert any additional drawings, storyboards, whiteboard pictures, project schedules, tasks lists, etc. that support your approach, design, and project. If you have no supporting documentation, please explain the rationale for why you are able to leave this section as N/A.*