**Project Plan**

* **Introduction**: This project will be creating a management system for use by public libraries. This system will allow users to interact with a clean and competent GUI to add, modify, and delete books with unique ISBN numbers within a database. Users can also check books in and out of the system, and search for books within the database to check descriptions and availability.
* **Project Organization:**

|  |  |  |
| --- | --- | --- |
| **Team Members** | **Roles of the members** | **Description** |
| Ethan Wheeler | Leader | Organizes and conducts meetings, planning, frontend Python coding |
| Nathan Hughes | Coding | Frontend Python support, possible backend support |
| Madison Schultz | Support | Python frontend coding, debug support |
| Brayden Orr | Coding | Backend coding |

* **Risk Analysis**:

|  |  |  |
| --- | --- | --- |
| Risk | **Probability of risk** | **Effects/Counters** |
| Project scope ends up outside the skill set of the team | High | The team will need to quickly adapt and use agile methods to implement task segments. Learning as the project progress is expected. Deficiencies can be lessened by using campus tutoring services. |
| The time required to develop the software is limited | Moderate | Eight weeks is relatively short for any significant program. However, the project was chosen specifically to meet requirements without being too complex. |
| The code generated by software code may be inefficient | Moderate | Python’s just-in-time compilation is known for being inferior to traditional compilers. It is not known how the project would theoretically scale in an enterprise environment. |
| software compatibility with older systems, not just new systems | Low | Python is one of the most supported and popular languages worldwide. There is little concern for low-power legacy systems in running the application. |

* **Hardware and Software requirements:**

The project will be using the current version of Python and its default libraries. All of the frontend script will be written in Python to create the application’s logic, while HTML will be used to construct the GUI that the end user will interact with. The backend logic will be bound in a Python DAO that interfaces with a SQLite database, which will handle book tracking. Generally, these software pieces will run on any remotely modern machine that was made in the last fifteen years. All of the dependencies used will be free software that anyone can download.

**Work Breakdown:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ACTIVITY** | **START** | **DURATION** | **START** | **DURATION** | **COMPLETE** |  | **PERIODS** |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **Planning** | 1 | 1 | 1 | 1 | **25%** |  |  |  |  |  |  |  |  |  |
| **Implementation** | 2 | 1 | 2 | 1 | **100%** |  |  |  |  |  |  |  |  |  |
| **Coding: Book Class – User Class** | 3 | 3 | 3 | 3 | **35%** |  |  |  |  |  |  |  |  |  |
| **Coding: Call Methods** | 3 | 3 | 3 | 3 | **10%** |  |  |  |  |  |  |  |  |  |
| **Coding: GUI interface** | 4 | 3 | 4 | 3 | **85%** |  |  |  |  |  |  |  |  |  |
| **Coding: Database/DAO Setup** | 4 | 3 | 4 | 3 | **85%** |  |  |  |  |  |  |  |  |  |
| **Coding: Admin Class** | 5 | 3 | 5 | 4 | **50%** |  |  |  |  |  |  |  |  |  |
| **Coding: Check-in/Check out** | 5 | 2 | 5 | 8 | **60%** |  |  |  |  |  |  |  |  |  |
| **Coding: Search Function** | 6 | 2 | 6 | 9 | **60%** |  |  |  |  |  |  |  |  |  |
| **Coding: Security/Validation** | 6 | 2 | 6 | 9 | **75%** |  |  |  |  |  |  |  |  |  |
| **Debugging/Troubleshooting** | 6 | 2 | 6 | 5 | **100%** |  |  |  |  |  |  |  |  |  |

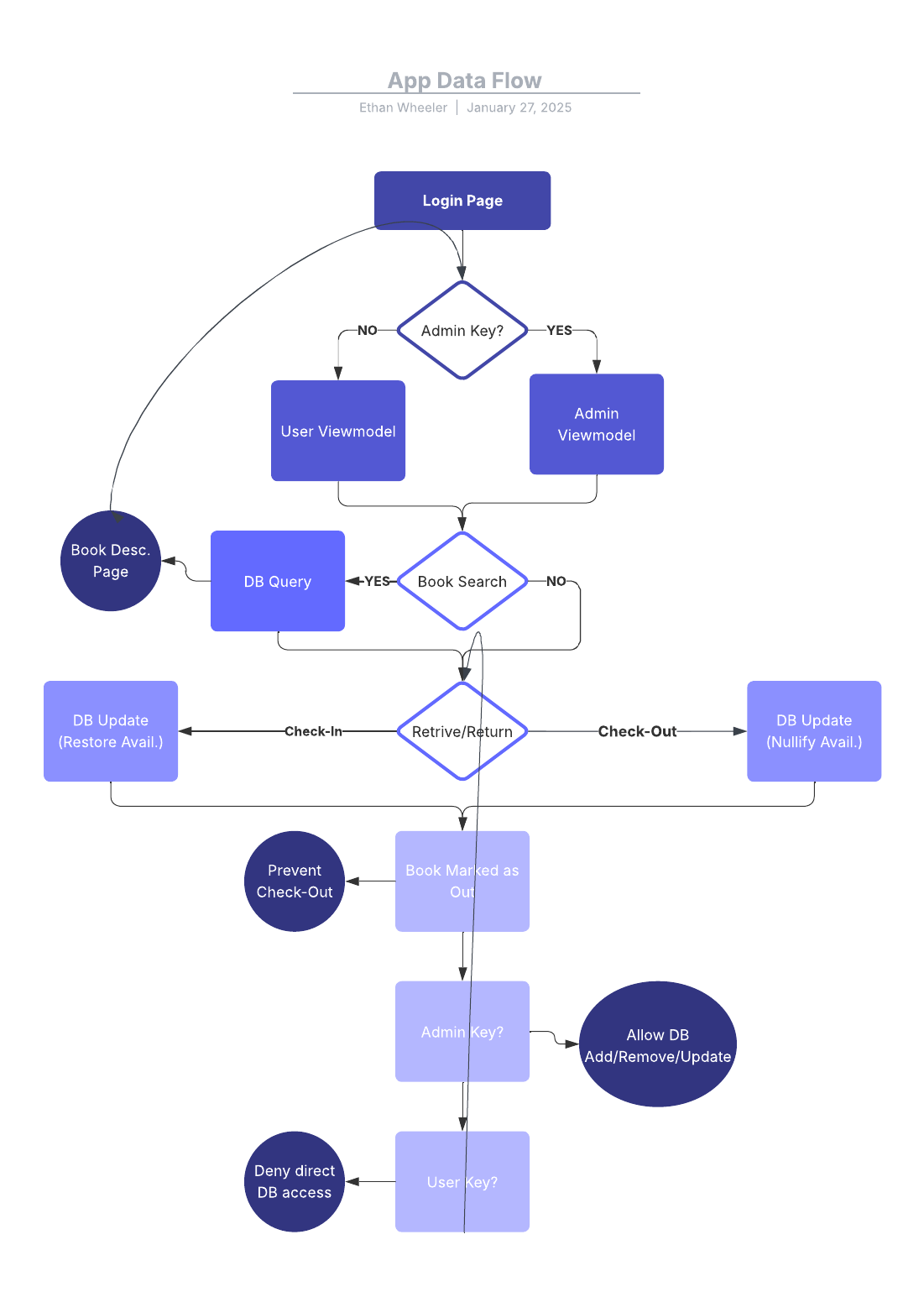
**Project schedule:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Effort(person-weeks)** | **Duration (weeks)** | **Dependencies** |
| **(T1)Planning (Wheeler, complete)** | 1 | >1 |  |
| **(T2)Implementation (Wheeler, complete)** | 1 | >1 | T1 |
| **(T3)Coding: Book Class – User Class (Hughes, Wheeler, complete)** | 2 | 1 | T2 |
| **(T4)Coding: Call Methods (All members, completed)** | 1 | 1 | T2 |
| **(T5)Coding: GUI interface (Hughes, completed)** | 1 | 1 | T2 |
| **(T6)Coding: Database/DAO Setup (Orr, Hughes, completed)** | 3 | 1 | T2 |
| **(T7)Coding: Admin Class (Not completed)** | 1 | 1 | T3-T6 |
| **(T8)Coding: Check-in/Check out (Schultz, Hughes, completed)** | 1 | 1 | T3-T6 |
| **(T9)Coding: Search Function (Completed)** | 3 | 1 | T3-T6 |
| **(T10)Coding: Security/Validation (Hughes, completed)** | 4 | 1 | T3-T6 |
| **(T11)Debugging/Troubleshooting (All members, completed)** | 2 | >1 | T3-T10 |

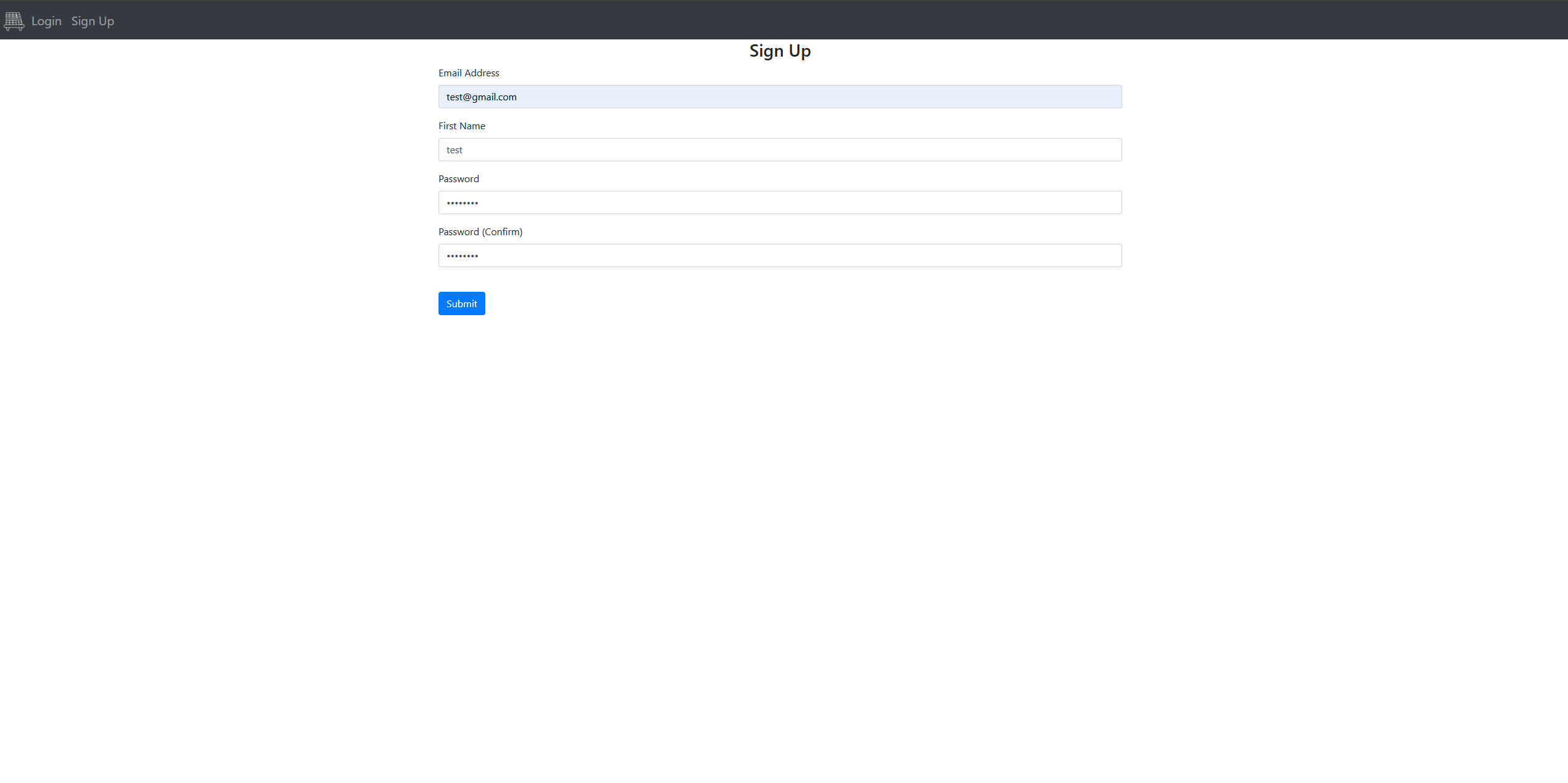
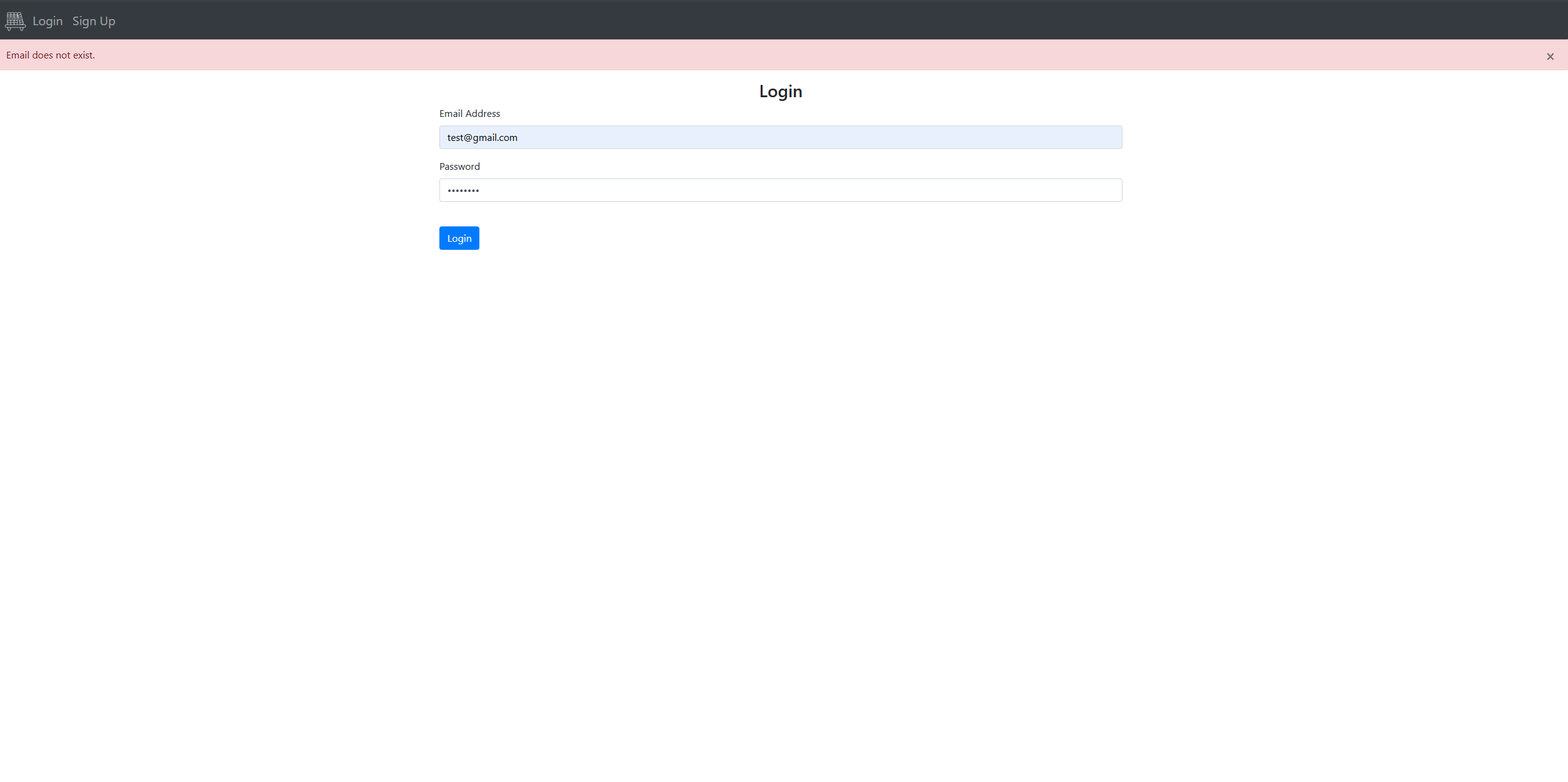
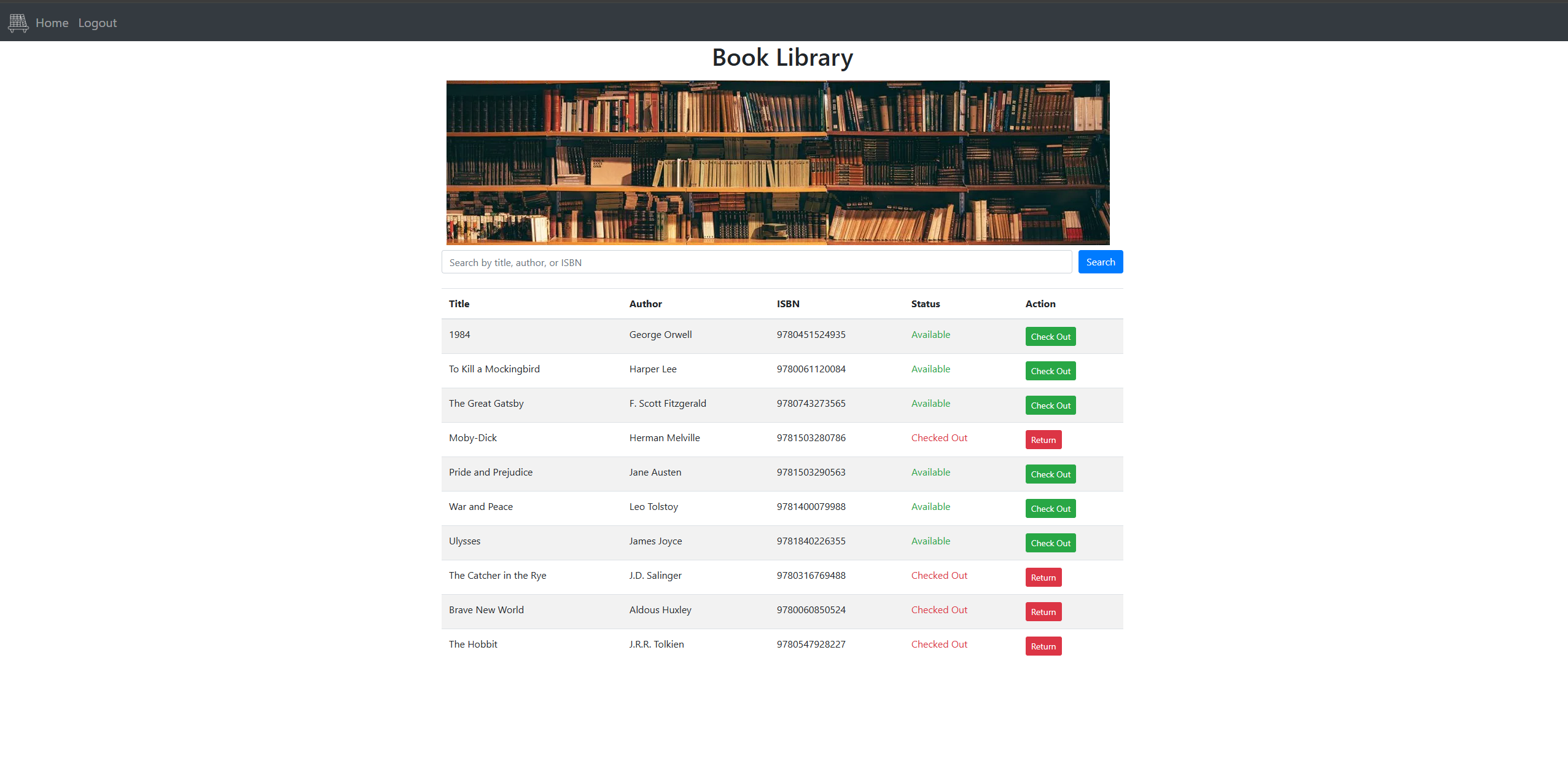
**Monitoring and Reporting:**

The project will be stored on a Github repository under the Leader’s account. The team will have access to any commits of the projects, along with keeping commits regular to ensure good checkpoints if something goes wrong.

**Appendix:**



**Test Cases:**

* Test to see if sign-up function works as intended
* Result: The viewmodel is successfully able to write to the database with a new user
* Test to see if the system will allow for an invalid email/password combination
* Result: System rejects the login attempt, with a pop-up message proclaiming the error to the user. The user can attempt to try their credentials again, if there was a typo.
* Test to see if the home screen works as intended.
* Result: The home page displays all of the entries in the backend, along with their availability status. The user can click on the green/red buttons to perform checkin/out functions.
* Test various parts of the search bar; test to see if it responds to various type of queries
* Result: The search bar can look up books via the ISBN, the name, or the author. The results are displayed as a lone entry, and still takes the same checkin/out inputs as the usual home screen does.

