**Document 10 – Sprint 2 Report**

This document is contained in your GitHub repository in a folder named *docs*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Group | 4 | | Group Member Names |  |
|  | |  |  | | --- | --- | | 1. | Emily Shirley | | 2. | Alfred Stanley | | |  |  | | --- | --- | | 3. | Merritt Hancock | | 4. | Mat Davidson | |
|  |  |  |

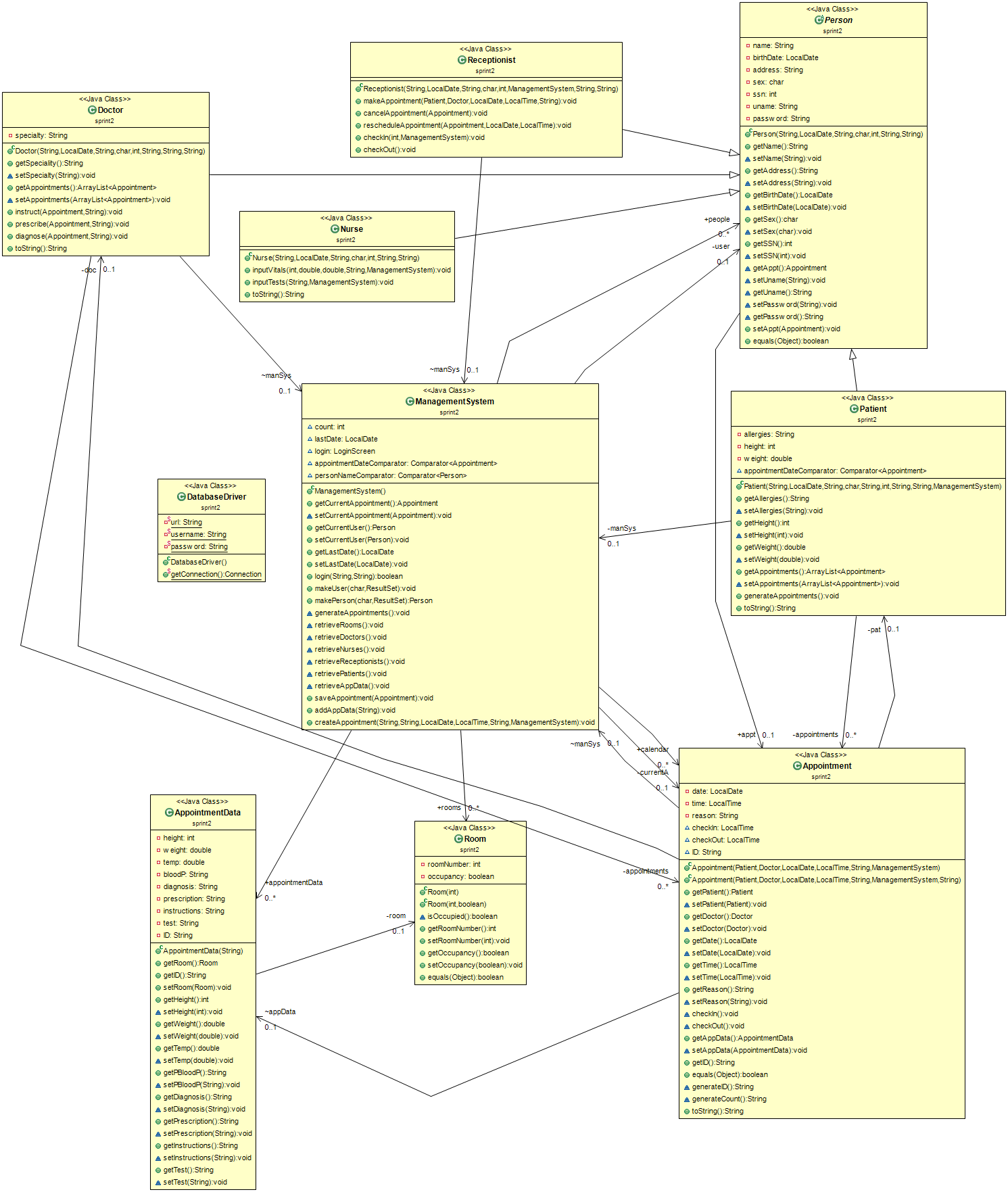
1. **Development Status**

|  |
| --- |
| **Deliverable**  Provide a numbered list of use cases that are complete and a brief description. |

|  |  |  |
| --- | --- | --- |
| **Num** | **Use Case Title** | **Description** |
| 1. | Create Appointment | A receptionist or doctor can create an appointment for a given patient. |
| 2. | Cancel Appointment | A receptionist can cancel a selected appointment from the appointment list. |
| 3. | Reschedule Appointment | A receptionist can reschedule an appointment by changing the date and time associated with it. |
| 4. | Prescribe Medications | A doctor can enter a prescription for the patient of the selected appointment. |
| 5. | Check-in | Patient gets checked into a room given by the receptionist currently logged in. |
| 6. | Check-out | Receptionist releases room that patient occupied. |
| 7. | Input Vitals | Nurse can input collected vitals such as blood pressure, height, weight, and temperature into the patient’s appointment data. |
| 8. | Input Test | A nurse can input a test needed for a patient of the selected appointment such as blood test. |
| 9. | Give Instructions | A doctor can enter instructions for the patient of the selected appointment. Instructions can include to rest. |
| 10. | Diagnose | A doctor can enter a diagnosis for the patient of the selected appointment. |
| 11. | Log-in | Users of the system need to provide credentials to gain access to sensitive information. |
| 12. | Log-out | When the user is finished with the system, they can log out so that operations cannot be performed under their account. |

1. **Class Diagram**

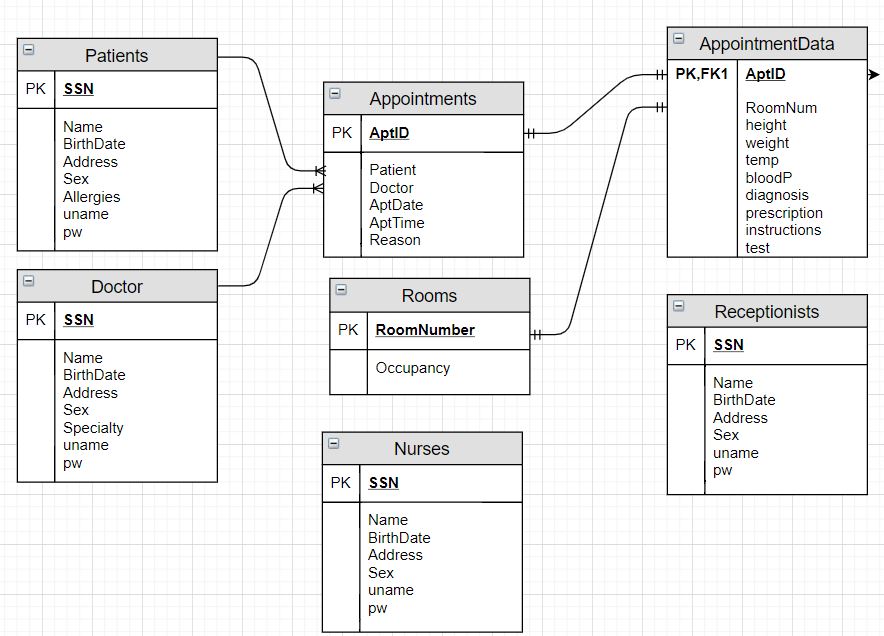
|  |
| --- |
| **Deliverable**  Develop a neat, legible, properly sized UML class diagram(s) showing your current design that exactly reflects your code. If it is too large to effectively display on a single page, then do either or both of these: (a) print on larger paper in the media center, (b) Show a class diagram that only shows associations among classes (no attributes or methods) and then have several diagrams that show several classes with associations, attributes, and methods. |



1. **Data Persistence Summary**

|  |
| --- |
| **Deliverable**  Explain how data persistence was implemented; format for files you used, or an ER diagram for a database; classes to do data persistence including a class diagram and how they are used. **This should be done in paragraph form.** |

We decided to use Google Cloud Platform SQL as a database. We used JDBC Connector/J in Eclipse for our database operations. We created a static DatabaseDriver class to establish connections. From there, Person subclass objects and the ManagementSystem would query the database to perform their respective operations. When the ManagementSystem is first created, it queries each table in the database to retrieve and create all Person, Appointment and AppointmentData objects and stores them locally. This way, the database only needs to be queried when a table needs to be updated.



1. **Retrospective**
2. Meet as a group and discuss the following questions and provide a group written response below:
3. Overall, how was the process of developing our system during Sprint 2 compared to Sprint 1?

|  |
| --- |
| **Answer**  We continued meeting weekly which helped us stay on track and make sure tasks were getting completed regularly. Once the GUI became the focus of the project, it became much more complex. Linking the GUI to our system and the system to our database took more considerations than we realized when planning in Sprint 1. |

1. What did not work well for us (ones continued from Sprint 1 and/or new ones)?

|  |
| --- |
| **Answer**  We sometimes ran into Git conflict errors as a result of occasionally not announcing what we were currently working on. We also may have waited a bit late to decide to go in the database direction for data persistence.  From time to time, we would have multiple people working on the same file or files dependent upon changes that were not yet pushed to the repository. Ultimately, communication was lacking at times. |

1. What actions can we take to improve our process going forward?

|  |
| --- |
| **Answer**  We could spend more time delegating tasks to ensure they get completed, and to track where issues may have arisen if they occur. Additionally, we could utilize the project issues on github more effectively to keep track of what has been done and what needs to be done. |