Team:
Matt Uhlar
Alex Goodwin
Lucy Wilkinson
Dmytro Ryzhkov

Title: 13_Athlete-Tracker

Features Implemented From Part 2:
Note: Rows with blank ID cells represent features that were not listed in a Part 2 requirement, but evolved and were implemented as we implemented the other requirements.

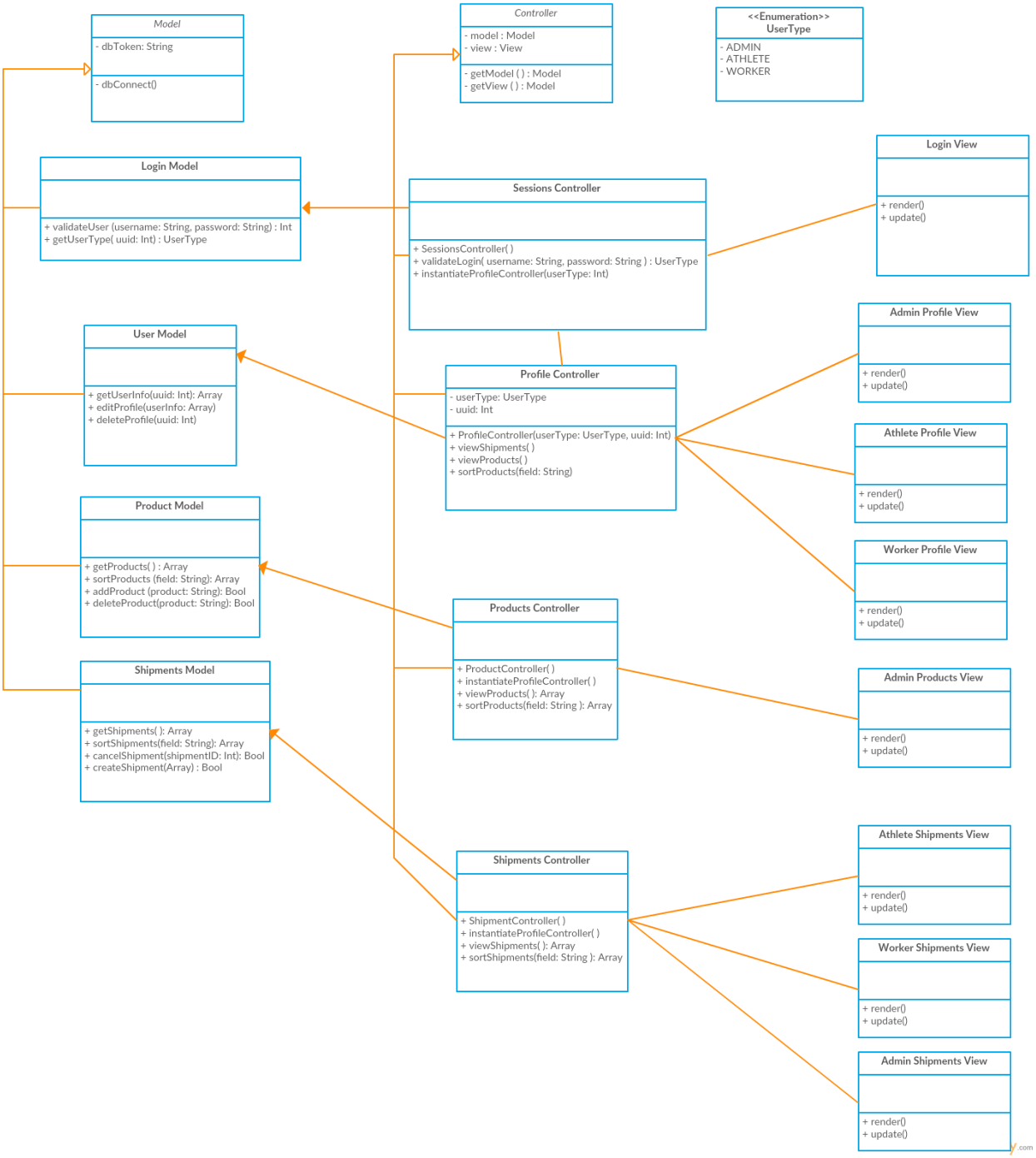
ID	Title
FR-02	Users can login to the system using a set username and password
	Users can logout
	Admins can view products
	Admins can filter products by status
BR-02	Admins can add new products
BR-03	Admins can edit products
	Admins can view athletes
	Admins can filter athletes by status
	Warehouse workers can view athletes
	Warehouse workers can filter athletes by status
BR-05	Admins can add new athletes
BR-06	Admins can edit athletes
	Admins can view admins
	Admins can filter admins by status
	Admins can add new admins
	Admins can edit admins
	Admins can view warehouse workers

	Admins can filter warehouse workers by status
	Admins can add new warehouse workers
	Admins can edit warehouse workers
BR-08	Admins can view a list of shipments
UR-01	Admins can view the value of a shipment
	Admins can filter shipments by status
BR-09	Athletes can view a list of shipments sent to them
UR-02	Athletes can view the value of a shipment
BR-10	Warehouse workers can view a list of shipments
BR-11	Admins can add and edit shipments
UR-06	All users can view and modify their information
UR-09	Warehouse workers can view shipments needing to be shipped
UR-10	Warehouse workers can update shipment status

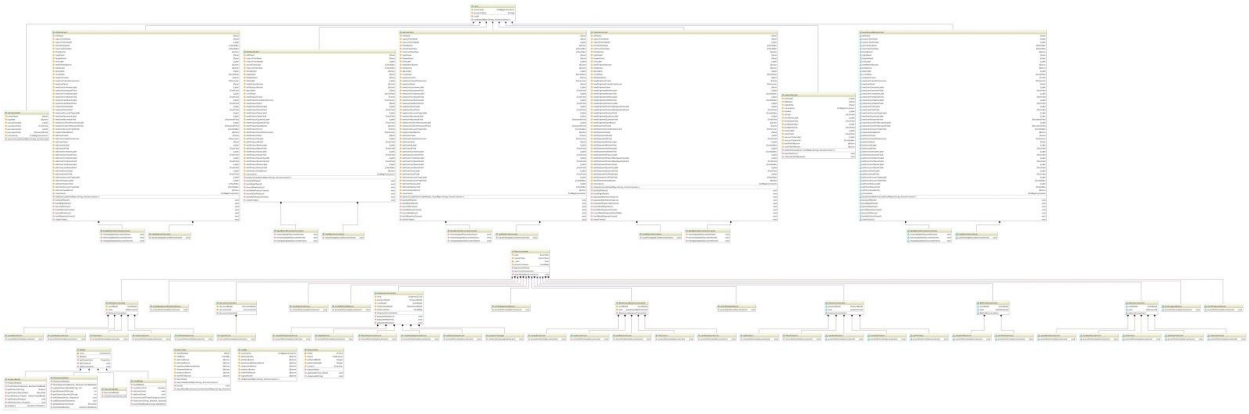
Features Not Implemented From Part 2:

ID	Title
UR-07	Admins can search for athletes by name
UR-08	Warehouse workers can search for athletes by name
UR-11	Users can retrieve lost passwords and verify identity
FR-01	System can calculate total value of shipments sent to an athlete over a period of time
FR-08	System can email users to reset password
NF-03	The portal should be accessible from any size device
NF-05	User passwords should be securely stored and accessed (salted & hashed)

Project Part 2 Class Diagram:



Final Class Diagram:



Full resolution version available at:
https://github.com/lucywilkinson/Athlete-Tracker/blob/master/13_Class-Diagram.jpg

One of the major differences from our Part 2 diagram to our final one is the inclusion of many more attributes, particularly in our Views. We decided to use GridBagLayout to format our GUI, so we needed a lot of elements to define the various panels, buttons, labels, and fields.

Additionally, we chose to use individual controllers for each type of user, rather than just one general user controller. This allowed us to have more variety in the options given for each user type.

We also realized that we did not need separate views for each type of user on each page. While we were unable to fully implement user-based permissions, we still incorporated these concepts into our views and left room to add this if we choose to work on this project in the future.

Design Patterns:
For our project, we decided to use the Model View Controller (MVC) design pattern, as shown in our class diagram. Each view is instantiated by a unique controller, which also queries the relevant model(s). The admins, athletes, and workers views and controllers all utilize the common User Model, while Products, Shipments, and Sessions have unique models. MVC allowed us to display the information from the database in different ways, depending on the context.

What have you learned about the process of analysis and design now that you have stepped through the process to create, design, and implement a system?:

Throughout this project we drastically improved our ability to program as a team. As we progressed through the project, communication and git organization improved, as did our ability to delegate related tasks among ourselves, rather than taking on large chunks individually.

We learned firsthand the time-saving benefits of creating class diagrams, sequence diagrams, use cases, and a thorough requirements/priority list. By agreeing on interface names beforehand, we were able to skip much group discussion and get straight to work coding.

We also learned how grouping common attributes and methods into parent classes and only overriding where necessary saves time by encouraging code reuse and interface standardization. Though some of this was learned the hard way or left with room for improvement, there is no doubt the experience of actually putting those themes into action as opposed to merely understanding them at a conceptual level deepened our understanding and appreciation for polymorphism and inheritance.

In addition, having mockups was certainly beneficial and allowed for rapid creation of views and a common understanding of what each view should contain and be able to do. If we had a stronger understanding of MVC from the start of the project, we could have established what should be in the model, view, and controller earlier, and avoided some of the confusion we encountered.