|  |
| --- |
|  |
| Week 13: Post Mortem |
| SDEV-450-81: Enterprise Java Programming |
| Instructor: Jason McGoff |
| **Students: Christopher Dierolf, Stephen Graybeal, Neil Hart, Thomas Muck** |
| **12/13/2019** |

|  |
| --- |
|  |

# 

# Introduction

The team was tasked with creating a multi-tiered J2EE application that included database interactivity and multi-threading. Our group was put together by default, with only four students in the course. After some initial discussion of different types of apps, we quickly decided on a Ticketmaster application that utilized the Ticketmaster API. We also decided to make it a desktop application, rather than a web-based app. We discussed potential features of the app, and a group of core features was decided on, as well as a few “nice to have” features that we could do if there was time at the end of the project. A user of the app would be able to search events, view a seating chart, purchase tickets, and receive tickets via email. We decided to use the Ticketmaster API to find events, but use a “hybrid” approach with venues – using the real venue name, but a fake seating chart to avoid complexities with large venues with tens of thousands of seats.

Christopher took the lead as the project manager and quickly got the group set up with Github, Slack, and Trello. Stephen set up a Google Drive to share documents for collaboration, and Tom set up an account with Amazon Web Services for a free SQL Server account. We decided to use a remote SQL Server instance rather than set up local servers on each computer, to more easily maintain the seating charts, purchased tickets, and to avoid complexities with installation of local servers.

# What Went Well

The team worked together cohesively as a unit, and each member took on tasks suited to his own strengths. There was enough overlap so that the members each learned about other parts of the app and the technologies involved as they worked on their own parts. The core part of the project came together quickly, and by the time we were ready to call it complete, it met our expectations. We were able to complete all of the requirements to our satisfaction, with time left over for testing.

# What Did Not Go Well

The biggest problem with the project was the fact that the team members were not familiar with the technologies required for creating a larger Java project. Our previous two courses in Java taught us the basics, but time was lost figuring out how to get required libraries to be part of the project. Some familiarity with a technology like Maven would have aided in that regard and saved us a lot of time. The workflow of GIT was also difficult to come to terms with at the beginning, and time was lost there, but after a meeting with the instructor, the team settled on a workflow that was more suited to an application of this size. There were still occasional hiccups, such as merge conflicts that we did not learn in any class, but Google was helpful there to show the required steps to resolve conflicts. Database connectivity and interactivity was another hiccup, because we did not have experience with connecting Java to a database, and what the best practices were in creating connections, ResultSets, and other JDBC technologies.

# What We Would Change

The project went well and there is not much that could have been done to resolve our inexperience, but perhaps if we had known about what some of the best practices are for building a Java project of this size, we could have saved some time. As a whole, however, the team is pleased with how the application came together, and we all learned a lot in the process.