CSE 308

Semester Project Discussion
Movie Info System (MIS)

Teams

- Aspen - Charles Bendernagel, Joseph Giardina, Konrad Juszkiewicz, and John Legutko
- Birch - Arthur Chan, Chun Maung, Ahsan Qureshi, and Bilawal Raja
- Cedar - ???
- Dogwood - ???

If enrollment stays at 17, there will either be 1 5-person team or a 2-person team and a 3-person team

Take time now to talk to other students and decide on your team
Movie Information System (MIS)

- The project is a re-engineering of a typical movie information system (e.g., Fandango)
- Initial requirements are in the existing system (some visible, some not), but you might expand and improve the system

One component of your grade on the project will be the “degree of difficulty” or number of completed use cases

MIS Components

- Web GUI (multiple roles)
- DB
- OO structure
- Persistence layer
- Business Logic
- Reports
- E-mail alerts
- Movie chain servers (e.g., Loew’s)
- Multi-server coordination
- Test data

Your first step in the project is to define the requirements

Some requirements you can derive from Fandango, others will be given to you

Next week we will discuss how to define these requirements
Basic Web GUI Requirements

- HTML 5
- Technologies (CSS, JavaScript, jQuery, etc.)
- Integrated text and graphics
- Mouse-over actions
- Dynamic updating
Database

- Supported CS database (e.g., MySQL)
- What are the entities?
- What are the attributes?
- What are the data types of the attributes?
- What data are not obvious from the GUI?
- What are the search fields?
- What searches are not obvious from the GUI (e.g., login)

OO Structure

- How do you identify the classes you need?
- How do you identify the attributes of each class?
- How do you determine the best type for an attribute?
- What is the best relationship among your classes?

Alert: there are more data structure interfaces than List and Array and more primitive types than String and int
Session 2 – Project

Persistence Layer

- Your design will be object-first
- Database design will allow you to persist much of your OO data
- You will implement a layer (or use a tool) to
  - Retrieve DB data
  - Persist OO data
- Requires OO-Relational mapping
- JPA strongly recommended

Business Logic

- Processing in your server in response to user requests, timers, external requests, etc.
  - For example, when to update the DB based on new movie chain schedules)
- How do you decompose a complex system into a limited set of system actions?
- How do you prioritize the business logic components so that you can develop a set of incremental builds?
Reports

- You will define the requirements for parts of the system that are not generally available (e.g., administrator maintenance)
- What reports are likely needed by administrators?

Servers

- Other servers coordinate with your system
- Examples
  - Movie Chains
  - Movie Production Studios
  - Ad servers
  - Payment processor
  - Geolocator
- You will develop minimal versions of the other systems (or use real systems) that will allow you to test your system
Multi-Server Coordination

- What is your interface mechanism between your MIS and the other servers?
  - E.g., how do you update your DB based on revised info?
- What formats do you support for interchange?

Test Data

- You need to develop or capture test data that will cover the spectrum of all the system possibilities
  - Movie data
  - Movie images
- You might need to develop a utility to capture movie data and load that data into your DB
Scope

- Scope of the system is defined by your set of use cases
- A recommended set of use cases will be given to you following the requirements phase
- You are required to implement all use cases on the list, but you might develop some additional ones

Approximate Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/13</td>
<td>List of use cases</td>
</tr>
<tr>
<td>2/27</td>
<td>GUl</td>
</tr>
<tr>
<td>3/6</td>
<td>Requirements (use cases, etc.)</td>
</tr>
<tr>
<td>3/10</td>
<td>Object design (initial class diagram)</td>
</tr>
<tr>
<td>Week of 3/20</td>
<td>Design reviews</td>
</tr>
<tr>
<td>3/27</td>
<td>Design document</td>
</tr>
<tr>
<td>4/3</td>
<td>Compilable code (stubbed objects)</td>
</tr>
<tr>
<td>4/3-4/10</td>
<td>Builds</td>
</tr>
<tr>
<td>4/10-4/17</td>
<td>Code reviews</td>
</tr>
<tr>
<td>4/24-5/3</td>
<td>Final demos</td>
</tr>
</tbody>
</table>
Requirements Gathering Steps

- Understand existing systems
- Identify areas of improvement
  - Business logic
  - User interface
- Define “non-visible” components of the system
- Propose improvements in your interface documents / presentations
- Consider limitations in scope

Components of System Development

- Requirements analysis
- Interface design
- Project plan
- Design (including design review)
- Builds (including code review)
- Test
- Documentation
- Demo

This will be important (and difficult)
Assignment

- Begin to browse the Fandango site to better understand the functions required
- Think about the more difficult parts of your system
- Full discussion of Fandango in class on Monday