Code Review Checklist

Coding Standards
- understandable
- adhere code guidelines
- indentation
- no magic numbers for testing
- naming
- units, bounds
- spacing: horizontal (btwn operators, keywords) and vertical (btwn methods, blocks)

Comments
- no needless comments
- no obsolete comments
- no redundant comments
- methods document parameters it modifies, functional dependencies
- comments consistent in format, length, level of detail
- no code commented out

Logic
- array indexes within bounds ✓
- conditions correct in ifs, loops ✓
- loops always terminate ✓
- division by zero ✓
- refactor statements in the loop to outside the loop

Error Handling
- error messages understandable and complete ✓
- edge cases (null, 0, negative) ✓
- parameters valid
- files, other input data valid

Code Decisions
- code at right level of abstraction
- methods have appropriate number, types of parameters
- no unnecessary features
- redundancy minimized
- mutability minimized
- static preferred over nonstatic
- appropriate accessibility (public, private, etc.) ✓
- enums, not int constants
- defensive copies when needed ✓
- no unnecessary new objects
- variables in lowest scope ✓
- objects referred to by their interfaces, most generic supertype ✓

Review Information:

Name of Reviewer: Lucas

Name of Coder: Hyde

File(s) under review: ScoreController.cs

Brief description of change being reviewed: _____

Review Notes (problems or decisions):

white space 15, 18, 33-36, 42, 54, 66, 87, 89, 92-94

unnecessary Comments 104-134

magic numbers for testing only

magic date on 97

SVN Versions (if applicable):

Before review: _____

After revisions: _____

Code Review Checklist

Coding Standards
- understandable
- adhere code guidelines
- indentation
- no magic numbers
- naming
- units, bounds
- spacing: horizontal (btwn operators, keywords) and vertical (btwn methods, blocks)

Comments
- no needless comments
- no obsolete comments
- no redundant comments
- methods document parameters it modifies, functional dependencies
- comments consistent in format, length, level of detail
- no code commented out

Logic
- array indexes within bounds
- conditions correct in ifs, loops
- loops always terminate
- division by zero
- refactor statements in the loop to outside the loop

Error Handling
- error messages understandable and complete
- edge cases (null, 0, negative)
- parameters valid
- files, other input data valid

Code Decisions
- code at right level of abstraction
- methods have appropriate number, types of parameters
- no unnecessary features
- redundancy minimized
- mutability minimized
- static preferred over nonstatic
- appropriate accessibility (public, private, etc.)
- enums, not int constants
- defensive copies when needed
- no unnecessary new objects
- variables in lowest scope
- objects referred to by their interfaces, most generic supertype
- Design
- Performance + Scalability
- More Testing

Review Information:

Name of Reviewer: Huda Syed

Name of Coder: Huda Syed

File(s) under review: ScoresController.cs

Brief description of change being reviewed: _____

Review Notes (problems or decisions):

- Line 15 → extra space
- Line 117 → what do these do
- Line 19 → what is this
- Line 33-35 → extra space
- Line 37 → what does this do
- Line 67 → do we really need this
- Line 87-89 → Are Extra
- Line 92-129 → Extra White Space
- Line 95 → what does this do.
- Line 71 → Should be score

SVN Versions (if applicable):
Before review: _____
After revisions: _____

Code Review Checklist

Coding Standards
- understandable
- adhere code guidelines
- indentation
- no magic numbers
- naming
- units, bounds
- spacing: horizontal (btwn operators, keywords) and vertical (btwn methods, blocks)

Comments
- no needless comments
- no obsolete comments
- no redundant comments
- methods document parameters it modifies, functional dependencies
- comments consistent in format, length, level of detail
- no code commented out

Logic
- array indexes within bounds
- conditions correct in ifs, loops
- loops always terminate
- division by zero
- refactor statements in the loop to outside the loop

Error Handling
- error messages understandable and complete
- edge cases (null, 0, negative)
- parameters valid
- files, other input data valid

Code Decisions
- code at right level of abstraction
- methods have appropriate number, types of parameters
- no unnecessary features
- redundancy minimized
- mutability minimized
- static preferred over nonstatic
- appropriate accessibility (public, private, etc.)
- enums, not int constants
- defensive copies when needed
- no unnecessary new objects
- variables in lowest scope
- objects referred to by their interfaces, most generic supertype

Review Information:

Name of Reviewer: Chris

Name of Coder: _____

File(s) under review: ScoreController.cs

Brief description of change being reviewed: _____

Review Notes (problems or decisions):

add comments to dumy data method
fully comment all code
complete error messages
create variable for DateTime/magic Num
remove old code
fix extra space
delete obsolete code

SVN Versions (if applicable):

Before review: _____

After revisions: _____