**Webapp Controllers**

* **AccountController**

1. **Login() [GET]**
   * **Returns: View()**
     + Displays the login page (default view associated with the action).
2. **Login(string email) [POST, async]**
   * **Returns:**
     + **View() – If:**
       - The email is null/empty/whitespace (with a model error and log).
       - The email is not a valid ETSU email address (adds error and logs warning).
     + **RedirectToAction("Index", "Home") – If:**
       - The email matches the ETSU email regex (successful login).
       - Also logs user login and page load time, and signs in the user using HttpContext.SignInAsync.
3. **Logout() [async]**
   * **Returns: RedirectToAction("Login")**
     + Logs the user out using HttpContext.SignOutAsync and redirects them to the login page.

* **GamesController**

1. **IndexAsync() [GET, async]**
   * **Returns: View(games)**
     + Renders the main games index page.
     + Dynamically retrieves a list of Game objects via *\_httpClient.GetGamesList().*
     + Adds play counts via *\_playCountManager.GetPlayCount(game.Id).*
     + Sorts the list by Id before returning.
     + Logs page load time and user activity.
2. **Play(int id) [GET, async]**
   * **Returns:**
     + **View(game) – if a game with the given id exists.**
       - Finds the game by ID, logs URL, increments its play count, and returns the game view.
     + **NotFound() – if no game with the given id is found.**
       - Logs a warning that the game was not found.
3. **Snake() [GET]**
   * **Returns: View()**
     + Returns the view for the deprecated Snake game page.
4. **Tetris() [GET]**
   * **Returns: View()**
     + Returns the view for the deprecated Tetris game page.

* **HomeController**

1. **Index() [GET, async]**
   * **Returns: View(\_httpClient.GetGamesList())**
     + Loads the (deprecated) home index page.
     + Passes in the full list of games from the MicroClient.
2. **Admin() [GET]**
   * **Returns: View()**
     + Displays the Admin page.
     + Logs the visit with the current user name.
3. **Privacy() [GET]**
   * **Returns: View()**
     + Displays the Privacy Policy page.
     + Logs the visit with the current user name.
4. **GameCriteria() [GET]**
   * **Returns: View()**
     + Displays the Game Criteria page (likely used to explain eligibility or rules).
     + Logs the visit with the current user name.
5. **TwoDotOne() [GET]**
   * **Returns: View()**
     + Displays the Version 2.1 page (possibly changelog or version-specific content).
6. **TwoDotTwo() [GET]**
   * **Returns: View()**
     + Displays the Version 2.2 page.
7. **TwoDotThree() [GET]**
   * **Returns: View()**
     + Displays the Version 2.3 page.
8. **TwoDotFour() [GET]**
   * **Returns: View()**
     + Displays the Version 2.4 page.
9. **Error() [GET]**
   * **Returns: View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier })**
     + Displays an Error page with diagnostic info.
     + Uses *Activity.Current?.Id* or *HttpContext.TraceIdentifier* to populate *RequestId*.

* **SnapshotsController**

1. **Index() [GET, async]**
   * **Returns: View(snapshots)**
     + Retrieves all saved snapshots via \_snapshotService.GetAllSnapshotsAsync().
     + Sets ViewBag.GitHash based on the environment.
     + Returns a view of the snapshot list.
2. **Create() [GET]**
   * **Returns: View()**
     + Displays the snapshot creation form (empty input view).
3. **Create(string description) [POST, async]**
   * **Returns: RedirectToAction(nameof(Index))**
     + Constructs a new Snapshot object with:
       - Current UTC time
       - Git commit hash (based on environment)
       - Current game play counts
       - Log file contents from Logs/ directory (if present)
     + Saves the snapshot using \_snapshotService.SaveSnapshotAsync().
     + Redirects back to the snapshots index.
4. **Details(string id) [GET, async]**
   * **Returns:**
     + **View(snapshot)** – if found.
     + NotFound() – if the snapshot with the given ID does not exist.
     + Also sets ViewBag.GitHash for displaying environment commit info.
5. **Rollback(string id) [POST, async]**
   * **Returns:**
     + **RedirectToAction(nameof(Index))**
     + Retrieves a snapshot and attempts to rollback play counts using PlayCountManager.RollbackToSnapshot().
     + Logs success or failure.
     + Sets *TempData["SuccessMessage"]* or *TempData["ErrorMessage"]* for display.
6. **GetGitCommitHash() [private method]**
   * **Returns: string – the current Git commit hash**
     + Executes a local shell command: git rev-parse HEAD.
     + Used internally in Index(), Details(), and Rollback() if running in development.