### ****1.List of Contributors****

* Berkhan Öztürk
* Batuhan Karaahmet
* Elif Naz Özgür

### 2. ****Task Matrix****

| **Task** | **Contributor** |
| --- | --- |
| Test implementation | Berkhan Öztürk |
| Test execution | Elif Naz Özgür |
| Log collection | Elif Naz Özgür |
| Report writing | Batuhan Karaahmet |
| Bug analysis & documentation | Batuhan Öztürk |

# PA3 - Acceptance Criteria and Test Results

## Acceptance Criteria and Corresponding Test Results

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Acceptance Criteria | Test Class | Result |
| Fetch player data from API | The application must fetch player data from the Premier League using the API and correctly deserialize player objects. | ApiServiceTest | ✅ Passed (3/3) |
| Filter players by name | Given a player list, filtering by name should return a matching list with case-insensitive matching. | FilterUtilsTest | ❌ Failed (1/3) |
| Filter players by position | Filtering by position (e.g., “Forward”) should return a list of players with exact position match. | FilterUtilsTest | ❌ Failed (1/3) |
| Player UI creation | UI elements must correctly generate player buttons with valid player data and layout. | PlayerButtonTest | ✅ Passed (4/4) |
| Squad formation panel generation | The panel must create the correct layout and player button slots according to the selected formation. | FormationUITest | ✅ Passed (3/3) |
| Player object integrity | The Player class must store and return correct data (ID, name, position, age, nationality, photo). | PlayerTest | ✅ Passed (3/3) |

### 4. ****Test Results Analysis****

#### ApiServiceTest

* **Status**: Passed
* **Analysis**: All tests for team and player fetching worked correctly. API responses were handled gracefully, including test coverage for invalid responses.
* **System Output** includes valid team IDs and player data.

#### FormationUITest

* **Status**: Passed
* **Analysis**: All GUI formation panels were created without errors. Layout logic for formations was verified, including edge cases for unknown formations.

#### PlayerButtonTest

* **Status**: Passed
* **Analysis**: All UI tests involving layout, size, player switching, and image loading ran successfully. No crashes or layout issues.

#### PlayerTest

* **Status**: Passed
* **Analysis**: The Player model’s getters/setters and toString() method are functioning as expected.

#### FilterUtilsTest

* **Status**: **Failed 1/3**
* **Issue**: testFilterByPosition() failed.
  + **Expected**: 1 player returned
  + **Actual**: 0 players returned
  + **Root Cause**: Position filtering uses Turkish labels, but test data may not match position mapping in FilterUtils.POSITION\_MAP.
  + **Potential Fix**: Ensure test player has matching Turkish label like "Forvet" instead of already-mapped English label "Attacker".

### 5. ****Logs and Outputs****

#### From ApiServiceTest

* **System.out** contains printed player data:

python-repl

KopyalaDüzenle

Alex Telles - Defender (Brazil), Yaş: 33, Fotoğraf: ...

David de Gea - Goalkeeper (Spain), Yaş: 35, Fotoğraf: ...

...

* **System.err**:

vbnet

KopyalaDüzenle

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation.

This is a harmless logger warning and can be ignored.

#### From FilterUtilsTest

* **AssertionFailedError**:

yaml

KopyalaDüzenle

expected: <1> but was: <0>

Indicates mismatch between expected and actual filtered player count.

### 6. Conclusion

This project introduces **Squad Master,** a Java-based desktop application designed to simulate squad creation using real football player data. The project integrates API data from RapidAPI, and offers GUI features for player selection, filtering, and formation design. The testing process evaluated API interactions, data filtering accuracy, user interface behavior, and domain model consistency. All core tests passed except one related to positional filtering, which revealed a mismatch in language mapping. The study suggests localized data consistency is vital for user-centric applications. Overall, the project demonstrates successful integration of asynchronous APIs with Java Swing UI design.