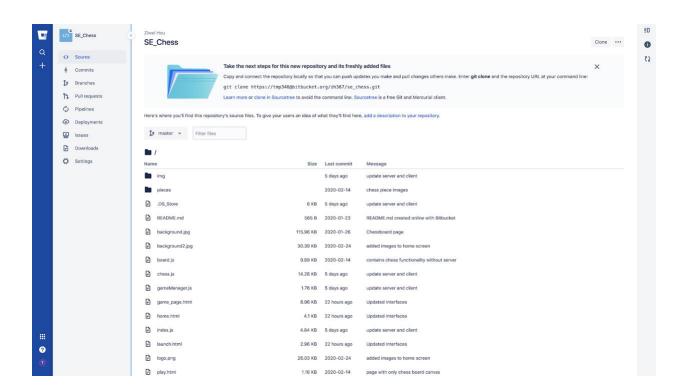
Software Tools

Version Control

We use BitBucket as our version control software. The software allows us to work on the same repository at the same time and see the newest updated. We can also revert the changes to the previous commit and merge conflict changes.



Static Analysis Tool

For the static analysis tool, we use ESLint which is a famous static analysis tool for JavaScript. It yields error on not only the syntactical error but also the stylistic error which helps us a lot in spotting those errors.



Figure 2: ESLint example error report

Test Running Tool

We use Mocha for our test running tool. It gives us a rich set of interfaces for asserting different objects and run our test cases.

```
Individual Components Unit Test
   Room Manager Test
     ✓ Create new user1, user2, user3
✓ Authenticate unregistered user_id
      ✓ Authenticate null user_id
       ✓ Authenticate created users
      ✓ User1 gets room, game should in the pause position
✓ User1 creates rejoins the created room
      ✓ User2 joins room with user1, game should be ready to start
✓ User3 joins the room with user1 and user2, this should be invalid

    ✓ User3 creates a new room, should be different than then room1
    ✓ User1 and user2 creates another room after creating a room
    ✓ User1 and User 2 joins another room after creating a room

       ✓ Unanthenticated user creates room
       ✓ Unanthenticated gets room
      ✓ Unanthenticated user joins an invalid room 
✓ Unanthenticated user joins a valid room
      ✓ Unanthenticated user quits room
       ✓ User quits room
   Game Manager Test
      ✓ User1 makes a valid move in their turn
✓ User1 makes a valid move, but not in their turn
     ✓ User2 makes a valid move, in their turn
✓ User2 makes a valid move, in their turn
✓ User1 makes a move in an invalid room
✓ User3 makes a valid move, in their turn, but the game is paused
  Chess API Test
      ✓ The default board should have 8x8 size
         The default board should have 32 chess pieces

    ✓ Test the default move history
    ✓ Test the default game status
    ✓ Test position of input pieces in the board

          Test out of boundary move
       ✓ Test invalid move
      ✓ Test empty postion move✓ Test move in the wrong turn✓ Test reverse to last move
          Test valid move
       ✓ Test game status when in check
      ✓ Test stalemate (draw game)
✓ Pawn Moves 2 steps forward at the first time
          The pawn moves two steps forward after the first time doing that, or tires to move to the positions which are not valid
       ✓ The pawn cannot make a move forward if there's no piece blocking the way
      ✓ The pawn cannot a move forward if there's a piece blocking the way
✓ The user's pawn attacks the other player's pieces
✓ The user's pawn cannot attack his own pieces
✓ The rook moves forward, backward, left, and right at any valid steps
✓ The user's rook attacks the other player's pieces
      ✓ The bishop moves diagonally at any valid steps
✓ The user's bishop attacks the other player's pieces
✓ The user's bishop cannot attack his own pieces
          The knight moves two squares vertically and one square horizontally, or two squares horizontally and one square vertically

    The user's knight attacks the other player's pieces
    The user's knight attacks his own pieces
    The queen should moves forward, backward, left, right, and diagonally at any valid steps
    The user's queen attacks the other player's pieces

          The user's queen cannot attack his own pieces
       √ The king should moves forward, backward, left, right, and diagonally at one step
√ The user's king attacks the other player's pieces
```

Figure 3: The unit tests for the project

```
Integration Test using Request Handler
  Rooms and Users related tests
   ✓ Create new user1, user2, user3, user4
    ✓ Authenticate unregistered user_id
   ✓ Authenticate null user_id
   ✓ Authenticate created users
   ✓ User1 creates a room
   ✓ User1 gets room, game should in the pause position
   ✓ User1 creates rejoins the created room
   ✓ User2 joins room with user1, game should be ready to start
   ✓ User3 and user4 try to join a full room
    ✓ User3 creates a new room, should be different than then room1
   ✓ User1 and user2 creates another room after creating a room
   ✓ User1 and User 2 joins another room after creating a room
   ✓ Unanthenticated user creates room
   ✓ Unanthenticated gets room
   ✓ Unanthenticated user joins an invalid room
   ✓ Unanthenticated user joins a valid room
   ✓ Unanthenticated user quits room
    ✓ User quits room
    ✓ User quits room without enter a room
  Chess move related tests
   ✓ User1 makes a valid move in their turn
   ✓ User1 makes a valid move, but not in their turn
   ✓ User2 makes a invalid move, in their turn
   ✓ User2 makes a valid move, in their turn
   ✓ User3 makes a valid move, in their turn, but the game is paused
   ✓ User4 (did not join any room) makes a move
   ✓ unauthenticated user makes a move
   ✓ User1 see if he/she is is_player1
   ✓ User2 see if he/she is is_player1
   ✓ User3 see if he/she is is_player1
   ✓ User4 see if he/she is is_player1
92 passing (62ms)
```

Figure 4: The integration tests for the project

Code Coverage Tool

We use Istanbul.JS for our code coverage tool. It gives us a summary table about the percentage of statements, branch, functions, and lines of code that our tests have covered.

File	 %	Stmts	 %	Branch	 %	Funcs	 %	 Klines	 Uncovered Line #s
			i		i		i		
All files		94.92	i	90	İ	100	i	94.75	
chess.js		95.33	1	89.06		100		95.16	75,77,79,102,105,117,127,183,205,297,361,426
gameManager.js		100	1	100		100		100	
requestHandler.js		100	1	100		100		100	
roomManager.js		91.18	l.	86.89		100	ı	91.09	44-46,51,71,164,173,186,193
					l		l		

Figure 5: The code coverage report table

Bug Tracking Tool

The BitBucket offers the issue and bug tracking tool that we can use to track and create any issue or bugs related to the project.

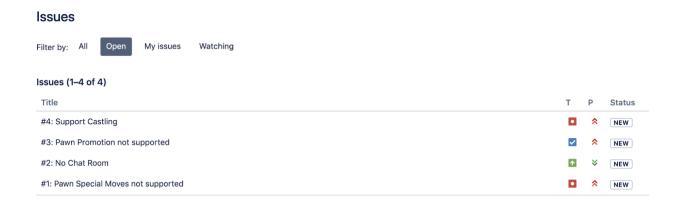


Figure 6: Bug tracking tool in BitBucket