

Experiment No - 09

Roll No.	52
Name	Kshitij Nangare
Class	D15B
Subject	Full Stack Development
Lab Outcome	L6
Date of Performance / Submission	13/10/2025 27/10/2025
Signature & Grades	

Experiment 9

Aim : CI/CD Deployment with GitHub Actions + Render/Vercel

Code :


```
backend >  app.js > ...
1  import dotenv from 'dotenv'; 7.2k (gzipped: 3.1k)
2  import express from 'express';
3  import cookieParser from 'cookie-parser'; 5k (gzipped: 2k)
4  import cors from 'cors'; 5k (gzipped: 2.1k)
5  import {connectDB} from './config/db.js';
6
7  import authRoutes from './routes/authRoutes.js';
8  import sessionRoute from './routes/sessionRoutes.js';
9
10 dotenv.config();
11
12 const app = express();
13 connectDB();
14
15 // Middleware
16 app.use(cors({ origin: process.env.FRONTEND_URL, credentials: true }));
17 app.use(cookieParser());
18 app.use(express.json());
19
20 app.use('/api/auth', authRoutes);
21 app.use('/api/session', sessionRoute);
22
23 // Server
24 const PORT = process.env.PORT || 3000;
25 app.listen(PORT, () => {
26   console.log(`Server running on port ${PORT}`);
27 });
28
```

Figure 1

```
backend > config > JS db.js > ...  
1  import mongoose from 'mongoose'; 581.7k (gzipped: 145.3k)  
2  import dotenv from 'dotenv'; 7.2k (gzipped: 3.1k)  
3  
4  dotenv.config();  
5  
6  export const connectDB = async () => {  
7    try {  
8      await mongoose.connect(process.env.MONGO_URI);  
9      console.log('MongoDB Connected...');  
10   } catch (err) {  
11     console.error('Database connection error:', err.message);  
12     process.exit(1);  
13   }  
14 };  
15
```

Figure 2

```

import mongoose from "mongoose"; 581.7k (gzipped: 145.3k)

const SessionSchema = new mongoose.Schema({
  user_id: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true
  },
  title: {
    type: String,
    trim: true
  },
  description: {
    type: String,
    default: ""
  },
  youtube_url: {
    type: String,
    trim: true
  },
  tags: {
    type: [String],
    default: []
  },
  status: {
    type: String,
    enum: ["draft", "published"],
    default: "draft"
  },
  imageUrl: {
    type: String,
    default: ""
  },
  likes: {
    type: Number,
    min: 0,
    default: 0
  },
  // NEW: Array to store IDs of users who liked this session
  likedBy: {
    type: [mongoose.Schema.Types.ObjectId],
    ref: "User", // Assuming you have a 'User' model
    default: []
  },
}, {
  timestamps: true // This will automatically add createdAt and updatedAt
});

const Session = mongoose.model("Session", SessionSchema);

export default Session;

```

Figure 3

```

export const getAllPublishedSessions = async (req, res) => {
  try {
    const { search } = req.query; // Extract search term from query parameters

    let query = { status: 'published' };
    if (search) {
      // Create a case-insensitive regex for title or tags
      const searchRegex = new RegExp(search, 'i');
      query.$or = [
        { title: { $regex: searchRegex } },
        { tags: { $in: [searchRegex] } } // Search within the tags array
      ];
    }

    const sessions = await Session.find(query)
      .populate('user_id', 'name email') // Populate creator details
      .sort({ createdAt: -1 }); // Sort by creation date, newest first

    res.json(sessions);
  } catch (err) {
    console.error("Error in getAllPublishedSessions:", err); // Log the error
    res.status(500).json({ message: err.message || "Server error" });
  }
};

// Get logged-in user's sessions
export const getMySessions = async (req, res) => {
  try {
    const sessions = await Session.find({ user_id: req.user._id })
      .populate('user_id', 'name email')
      .sort({ createdAt: -1 });
    res.json(sessions);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
};

```

Figure 4

```

export const getSessionById = async (req, res) => {
  try {
    const { id } = req.params; // Get the session ID from the URL parameters

    // Find the session and populate the user_id field to get creator's name and email
    const session = await Session.findById(id).populate('user_id', 'name email');

    if (!session) {
      return res.status(404).json({ message: 'Session not found.' });
    }

    res.status(200).json(session);
  } catch (error) {
    console.error('Error fetching session by ID:', error);
    // Handle specific CastError if ID format is invalid
    if (error.name === 'CastError') {
      return res.status(400).json({ message: 'Invalid session ID format.' });
    }
    res.status(500).json({ message: 'Server error.', error: error.message });
  }
};

// Create new session
export const createSession = async (req, res) => {

  const session = new Session({
    ...req.body,
    user_id: req.user._id
  });

  try {
    const newSession = await session.save();
    res.status(201).json(newSession);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
};

```

Figure 5

```

backend > controllers > sessionsController.js > ...
 97 export const likeSession = async (req, res) => {
 98   const userId = req.user._id; // auth middleware sets req.user.id
 99   const { id } = req.params; // Session ID from the URL parameter
100
101   try {
102     const session = await Session.findById(id);
103
104     if (!session) {
105       return res.status(404).json({ message: 'Session not found.' });
106     }
107
108     let updatedSession;
109
110     // Check if the user has already liked this session
111     if (session.likedBy.includes(userId)) {
112       // If already liked, UNLIKE IT: Decrement likes and remove user ID
113       updatedSession = await Session.findByIdAndUpdate(
114         id,
115         {
116           $inc: { likes: -1 }, // Atomically decrements the 'likes' count by 1
117           $pull: { likedBy: userId } // Removes the user's ID from the 'likedBy' array
118         },
119         {
120           new: true,
121           runValidators: true
122         }
123       ).populate('user_id', 'name email');
124       return res.status(200).json({ message: 'Session unliked successfully.', session: updatedSession });
125     } else {
126       // If not liked, LIKE IT: Increment likes and add user ID
127       updatedSession = await Session.findByIdAndUpdate(
128         id,
129         {
130           $inc: { likes: 1 }, // Atomically increments the 'likes' count by 1
131           $push: { likedBy: userId } // Adds the user's ID to the 'likedBy' array
132         },
133         {
134           new: true,
135           runValidators: true
136         }
137       ).populate('user_id', 'name email');
138       return res.status(200).json({ message: 'Session liked successfully.', session: updatedSession });
139     }
140   } catch (error) {
141     console.error('Error processing like/unlike for session:', error);

```

Figure 6

```

// Update a session
export const updateSession = async (req, res) => {
  try {
    const session = await Session.findOneAndUpdate(
      { _id: req.params.id, user_id: req.user._id },
      req.body,
      { new: true }
    ).populate('user_id', 'name email');
    if (!session) return res.status(404).json({ message: 'Session not found' });
    res.json(session);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
};

// Delete a session
export const deleteSession = async (req, res) => {
  try {
    const session = await Session.findOneAndDelete({
      _id: req.params.id,
      user_id: req.user._id
    });
    if (!session) return res.status(404).json({ message: 'Session not found' });
    res.json({ message: 'Session deleted' });
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
};

```

Figure 6


```

import User from '../models/User.js';
import { generateToken } from '../utils/jwtToken.js';
import bcrypt from 'bcryptjs'; 20.1k (gzipped: 9k)

export const register = async (req, res) => {
  try {
    const { email,name, password } = req.body;

    // Validate input
    if (!email || !password || !name) {
      return res.status(400).json({ message: 'Email and password are required' });
    }

    // Check if user exists
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(409).json({ message: 'Email already in use' });
    }

    // Hash password
    const salt = await bcrypt.genSalt(10);
    const password_hash = await bcrypt.hash(password, salt);

    // Create new user
    const user = new User({ email,name, password: password_hash });
    await user.save();

    // Generate JWT
    const token = generateToken(user);

    res.status(201).json({
      token,
      user: {
        id: user._id,
        name: user.name,
        email: user.email
      }
    });
  } catch (error) {
    console.error('Registration error:', error);
    res.status(500).json({ message: 'Registration failed' });
  }
};

```

Figure 7

```

export const login = async (req, res) => {
  try {
    const { email, password } = req.body;

    // Validate input
    if (!email || !password) {
      return res.status(400).json({ message: 'Email and password are required' });
    }

    // Find user
    const user = await User.findOne({ email });
    if (!user) {
      return res.status(401).json({ message: 'Invalid Email or Password' });
    }

    // Check password
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.status(401).json({ message: 'Invalid Email or Password' });
    }

    // Generate JWT
    const token = generateToken(user);

    res.json({
      token,
      user: {
        id: user._id,
        name: user.name,
        email: user.email
      }
    });
  } catch (error) {
    console.error('Login error:', error);
    res.status(500).json({ message: 'Login failed' });
  }
};

```

Figure 8

```

backend > controllers > authController.js > register
85 export const changePassword = async (req, res) => {
86   const { currentPassword, newPassword } = req.body;
87
88   // 1. Basic validation
89   if (!currentPassword || !newPassword) {
90     return res.status(400).json({ message: 'Please provide current password and new password.' });
91   }
92
93   if (newPassword.length < 4) {
94     return res.status(400).json({ message: 'New password must be at least 4 characters long.' });
95   }
96
97   try {
98     // req.user will come from your authentication middleware (e.g., JWT verification)
99     // It should contain the ID of the authenticated user
100    const user = await User.findById(req.user._id).select('+password'); // Select password field explicitly
101
102    if (!user) {
103      return res.status(404).json({ message: 'User not found.' });
104    }
105
106    // 2. Compare current password
107    const isMatch = await bcrypt.compare(currentPassword, user.password);
108
109    if (!isMatch) {
110      return res.status(401).json({ message: 'Current password is incorrect.' });
111    }
112
113    // 3. Check if new password is the same as current password
114    if (newPassword === currentPassword) {
115      return res.status(400).json({ message: 'New password cannot be the same as the current password.' });
116    }
117
118    // 4. Hash the new password and save
119    const salt = await bcrypt.genSalt(10);
120    user.password = await bcrypt.hash(newPassword, salt); // Hash with a salt round of 10
121    await user.save();
122
123    res.status(200).json({ message: 'Password changed successfully!' });
124
125  } catch (error) {
126    console.error('Error changing password:', error);
127    res.status(500).json({ message: 'Server error. Could not change password.' });
128  }
129 };

```

Figure 9

```

frontend > src > pages > Dashboard.jsx > ...
16
17 const DashboardPage = () => {
18   const { user, logout } = useAuthStore();
19   const [searchTerm, setSearchTerm] = useState('');
20   const [isLoading, setIsLoading] = useState(true);
21   const [activeTab, setActiveTab] = useState('all'); // 'all' or 'my'
22   const [allSessions, setAllSessions] = useState([]);
23   const [processingCardId, setProcessingCardId] = useState(null); // For like/unlike
24   const navigate = useNavigate();
25
26   // Function to fetch all published sessions
27   const fetchAllPublishedSessions = useCallback(async () => {
28     try {
29       setIsLoading(true); // Keep loading true for all sessions tab
30       // Append searchTerm as a query parameter for backend filtering
31       const response = await axiosInstance.get(`/api/session/get-all-sessions?search=${searchTerm}`);
32       setAllSessions(response.data);
33     } catch (error) {
34       console.error('Failed to fetch all published sessions:', error);
35       toast.error('Failed to load public sessions.');
```

Figure 10

```

frontend > src > pages > Dashboard.jsx > ...
17  const DashboardPage = () => {
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77  return (
78    <div className="min-h-screen bg-gray-50">
79      {/* Header */}
80      <Navbar />
81
82      {/* Main Content */}
83      <main className="max-w-7xl mx-auto px-6 py-8">
84
85        {/* Tab Navigation */}
86        <div className="border-b border-gray-200 mb-6">
87          <nav className="-mb-px flex space-x-8">
88            <button
89              onClick={() => setActiveTab('all')}
90              className={` whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm ${
91                activeTab === 'all'
92                  ? 'border-indigo-500 text-indigo-600'
93                  : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
94              }`>
95              All Sessions
96            </button>
97            <button
98              onClick={() => setActiveTab('my')}
99              className={` whitespace-nowrap py-4 px-1 border-b-2 font-medium text-sm ${
100                activeTab === 'my'
101                  ? 'border-indigo-500 text-indigo-600'
102                  : 'border-transparent text-gray-500 hover:text-gray-700 hover:border-gray-300'
103              }`>
104              My Sessions
105            </button>
106          </nav>
107        </div>
108
109        {/* Search Bar (Only for All Sessions tab) */}
110        {activeTab === 'all' && (
111          <div className="relative mb-6">
112            <div className="absolute inset-y-0 left-0 pl-3 flex items-center pointer-events-none">
113              <Search className="h-5 w-5 text-gray-400 aria-hidden="true" />
114            </div>
115            <input
116              type="text"
117              name="search"
118              id="search"
119            />
120          </div>
121        )}
122      </main>
123    </div>
124  )
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

Figure 11

```

frontend > src > pages > Dashboard.jsx > ...
17  const DashboardPage = () => {
123      value={searchTerm}
124      onChange={(e) => setSearchTerm(e.target.value)}
125      />
126  </div>
127  })
128
129  /* Sessions Display */
130  {activeTab === 'all' ? (
131      <section>
132          <h2 className="text-xl font-semibold text-gray-900 mb-4">All Wellness Sessions</h2>
133          {isLoading ? (
134              <LoadingSpinner />
135          ) : allSessions.length === 0 ? (
136              <div className="bg-white p-8 rounded-lg shadow-sm text-center">
137                  <p className="text-gray-500">No sessions found matching your search.</p>
138                  <p className="text-gray-400 text-sm mt-2">Try a different keyword or check back later!</p>
139              </div>
140          ) : (
141              <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
142                  {allSessions.map(session => ( // Use `allSessions` here, as filtering is done on backend
143                      <SessionCard
144                          key={session._id}
145                          session={session}
146                          isEditable={false} // Public sessions are not editable from here
147                          onLike={handleLikeUnlike}
148                          hasLiked={user && session.likedBy && session.likedBy.includes(user.id)} // Check if
149                          isProcessing={processingCardId === session._id}
150                      </>
151                  )}}
152              </div>
153          )}
154      </section>
155  ) : (
156      <section>
157          <MySessionsPage />
158      </section>
159  )}
160  </main>
161  </div>
162  );
163  };
164
165  export default DashboardPage;

```

Figure 12

```

sessionsController.js  vercel.json  authRoutes.js
frontend > vercel.json > ...
1  {
2      "rewrites": [
3          { "source": "/(.*)", "destination": "/" }
4      ]
5  }
6

```

Figure 13

Output :

The screenshot shows the Vercel Production Deployment page for the 'wellnest' project. The page is dark-themed and features a navigation bar at the top with links to Overview, Deployments, Analytics, Speed Insights, Logs, Observability, Firewall, Storage, Flags, and Settings. The main content area displays the 'Production Deployment' status, which is 'Ready'. A preview of the application is shown on the left, and deployment details are on the right. The deployment was created on Aug 1 by Roshan-504. The source is 'main' and the commit is 'ceaa378'. The deployment is for the 'wellnest-dterjueyf-roshan-yadavs-projects-4bce4824.vercel.app' domain. The page also includes buttons for 'Build Logs', 'Runtime Logs', and 'Instant Rollback'.

wellnest

Repository Usage Domains Visit

Production Deployment

Build Logs Runtime Logs Instant Rollback

Deployment

wellnest-dterjueyf-roshan-yadavs-projects-4bce4824.vercel.app

Domains

wellnest-kappa.vercel.app

Status Created

Ready Aug 1 by Roshan-504

Source

main

ceaa378 updated

Deployment Settings 3 Recommendations

Figure 14

The screenshot shows the Vercel Deployments page for the 'wellnest' project. The page is dark-themed and features a navigation bar at the top with links to Overview, Deployments, Analytics, Speed Insights, Logs, Observability, Firewall, Storage, Flags, and Settings. The main content area displays a list of deployments. The first deployment is 'BSJ941SA4' with status 'Ready' and 'Current'. The second deployment is 'BzMKv7Sxf' with status 'Ready'. The third deployment is '8RwUP3Cz2' with status 'Ready'. The fourth deployment is '7FYRsc7q8' with status 'Ready'. The fifth deployment is '7SLKFyS4e' with status 'Ready'. The page also includes a search bar and filters for 'All Branches...', 'All Authors...', 'All Environments', and 'Select Date Range'.

Deployments

Upgrade to Pro for 2x more CPUs and faster builds

Automatically created for pushes to Roshan-504/WellNest

All Branches... All Authors... All Environments Select Date Range Status 5/6

BSJ941SA4 Production Current	Ready 11s (72d ago)	main ceaa378 updated	Aug 1 by Roshan-504
BzMKv7Sxf Production	Ready 13s (74d ago)	main 94d6269 Live Demo added	Jul 31 by Roshan-504
8RwUP3Cz2 Production	Ready 13s (74d ago)	main 676d750 Live Demo added	Jul 31 by Roshan-504
7FYRsc7q8 Production	Ready 12s (74d ago)	main f25f51c Update .env	Jul 31 by Roshan-504
7SLKFyS4e Production	Ready 10s (74d ago)	main a12065e Readme added	Jul 31 by Roshan-504

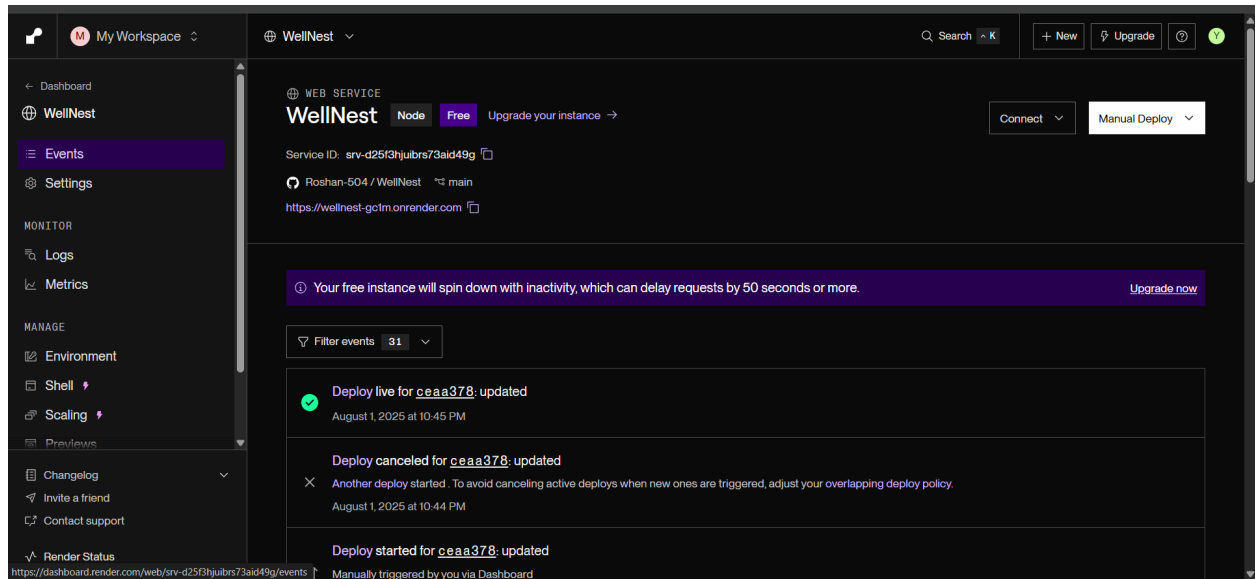


Figure 15

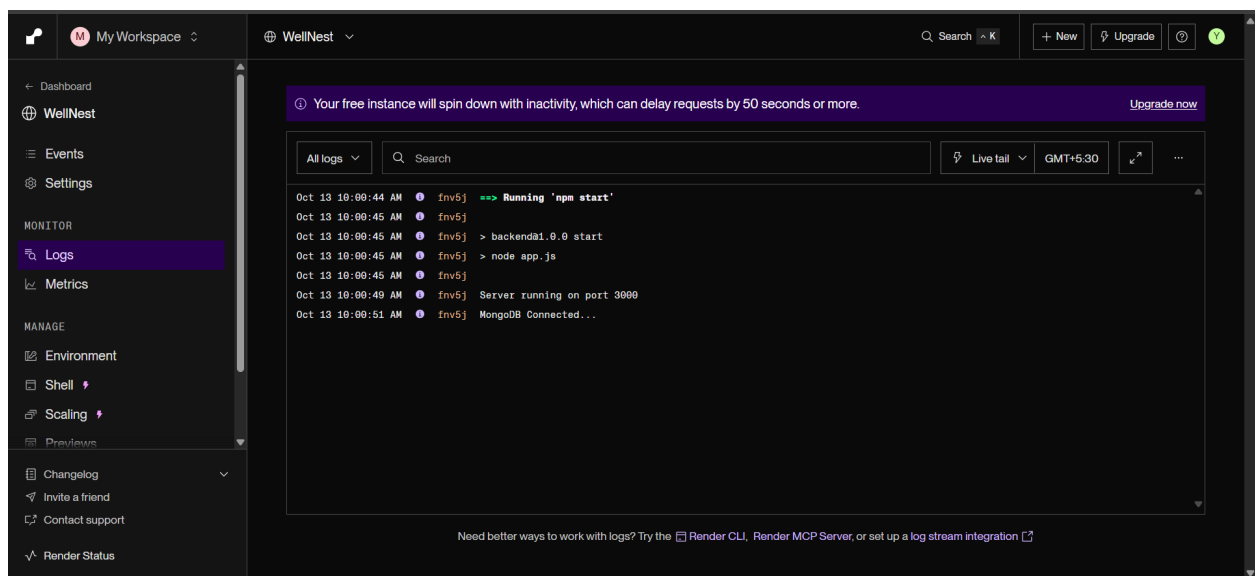


Figure 16