Blood Bank Automation with Bull: Complete Documentation

Grok 4 by xAI

August 30, 2025

Contents

1 Introduction

This document provides a complete, detailed summary and setup for the MERN Blood Bank automation using Bull. It covers what happens at each step, why Bull is used, and how the three cases work together. The content is structured for production-level understanding.

2 What is Bull and Why We Use It

Bull is a Node.js library for background jobs that uses Redis as a persistent job queue.

2.1 Purpose

Automate tasks that are:

- Long-running (PDF generation, email, SMS)
- Scheduled (daily scans)
- Multiple database updates
- Require retries if failed

2.2 Why not just cron

Cron (node-cron) only runs jobs in-process and can fail if the server restarts. Bull ensures persistence, retries, and concurrency, even if the Node server crashes.

2.3 Redis Role

Acts as a central store for jobs. Keeps jobs persistent and allows multiple workers to pick jobs reliably.

2.4 Worker Role

Background process that reads jobs from Redis and executes them. Can process multiple jobs concurrently.

3 Three Automation Cases

3.1 Case 1: Daily Near-Expiry Blood Report

Goal: Send admin a PDF report of blood bags/components nearing expiry.

Process:

1. Schedule job: Daily at 8 AM, a job is pushed to Bull queue (‘expiry-report‘).
2. Worker picks job: Background worker queries MongoDB for blood bags expiring within 5 days.
3. Generate PDF: Using PDFKit, a PDF report of near-expiry blood is created.
4. Send Email: Nodemailer sends the PDF to admin.
5. Logs/Completion: Worker logs success; if email fails, Bull can retry automatically.

Summary:

- No server blocking
- Automated daily report
- Reliable retries in case of failure

3.2 Case 2: Donors' Birthday SMS

Goal: Send automated birthday wishes to donors via SMS.
Process:

1. Schedule job: Daily at 9 AM, a job is queued (‘birthday-sms‘).
2. Worker picks job: Finds donors whose birthdays match today in MongoDB.
3. Send SMS: Using Twilio (or another SMS provider), sends personalized messages.
4. Concurrency: Can send multiple SMS at the same time without slowing server.
5. Retries: If an SMS fails, Bull can retry automatically.

Summary:

- Fully automated donor engagement
- Background task ensures main server performance
- Reliable for many donors

3.3 Case 3: Auto-Discard Expired Blood + Report

Goal: Automatically discard expired blood bags/components, generate a report, and email admin.
Process:

1. Schedule job: Daily at 7 AM, job queued in ‘discard-expired‘.
2. Worker picks job: Queries all expired blood bags/components across models. Marks them as ‘discarded‘ in MongoDB.
3. Generate PDF: Creates a report of discarded blood using PDFKit.
4. Send Email: Admin receives PDF via Nodemailer.
5. Retries: If any step fails (DB update, PDF, email), Bull retries automatically.

Summary:

- Complex multi-model operations done in background safely
- Admin always receives report
- No risk of missing tasks if server restarts

4 How Everything Works Together

1. Queues: ‘expiry-report‘ → daily PDF/email of near-expiry, ‘birthday-sms‘ → daily donor birthday SMS, ‘discard-expired‘ → daily discard + report.
2. Redis: Holds all jobs persistently, handles retries, failed jobs, delayed jobs.
3. Workers: Continuously running background Node process, picks jobs from Redis and executes logic.
4. Job Flow Example (Case 3): Job scheduled in Redis → Worker picks job → DB updates expired blood → Generate PDF → Send Email → Log success. All steps happen automatically without manual intervention.
5. PDF Generation: Uses PDFKit to create a human-readable report, saved to ‘/reports/‘ folder, can attach directly to email.
6. Email/SMS Sending: Emails via Nodemailer, SMS via Twilio, both can retry automatically if failed.

5 Advantages of This Approach

Feature	Benefit
Background Jobs	No blocking of API/server
Persistent Jobs	Safe from server crashes or restarts
Automatic Retries	Tasks are reliable even if failures occur
Concurrency	Multiple SMS/email or DB operations can run in parallel
Scheduling	Cron-style jobs for daily, weekly, hourly tasks
Centralized Queue (Redis)	Easy monitoring, can scale workers horizontally

6 Folder & Code Structure Example

```
bloodbank-app/

  models/
     BloodBag.js
     Donor.js

  queues.js         # Define all queues
  worker.js         # Process all jobs
  server.js         # Express API
  reports/          # Generated PDFs
  package.json
```

7 What Happens End-to-End Daily

- **7:00 AM** → Discard expired blood, generate report, send email
- **8:00 AM** → Generate near-expiry report, send email
- **9:00 AM** → Send birthday SMS

Bull + Redis + Workers ensures:

- Tasks run in background
- Tasks are persistent and reliable
- Admin receives reports automatically
- Donors receive birthday messages automatically

This setup makes your MERN Blood Bank fully automated and production-ready, handling multiple complex tasks reliably without blocking your server.

8 Setup Prerequisites

1. MERN stack ready (MongoDB, Express, Node.js, React optional for frontend)
2. Redis installed and running (‘localhost:6379‘ by default)
3. Node packages to install:

   ```
   npm install bull nodemailer pdfkit moment twilio
   ```

- ‘bull‘ → Job queues
- ‘nodemailer‘ → Sending emails
- ‘pdfkit‘ → PDF generation
- ‘moment‘ → Date handling
- ‘twilio‘ → SMS sending (optional, can use other providers)

9 Create 3 Separate Queues

```
const Queue = require('bull');

// 1. Daily near-expiry blood report
const expiryReportQueue = new Queue('expiry-report', 'redis://127.0.0.1:6379');

// 2. Donors birthday SMS
const birthdayQueue = new Queue('birthday-sms', 'redis://127.0.0.1:6379');

// 3. Auto-discard expired blood + generate report
const discardQueue = new Queue('discard-expired', 'redis://127.0.0.1:6379');

module.exports = { expiryReportQueue, birthdayQueue, discardQueue };
```

10 Add Jobs with Schedules

10.1 Expiry Report Queue

Runs daily at 8 AM:

```
const { expiryReportQueue } = require('./queues');

expiryReportQueue.add('send-expiry-report', {}, {
  repeat: { cron: '0 8 * * *' } // every day at 8 AM
});
```

10.2 Birthday SMS Queue

Runs daily at 9 AM:

```
const { birthdayQueue } = require('./queues');

birthdayQueue.add('send-birthday-sms', {}, {
  repeat: { cron: '0 9 * * *' } // every day at 9 AM
});
```

10.3 Discard Expired Blood Queue

Runs daily at 7 AM:

```
const { discardQueue } = require('./queues');

discardQueue.add('discard-expired', {}, {
  repeat: { cron: '0 7 * * *' } // every day at 7 AM
});
```

11 Worker Code (Processing Jobs)

11.1 Helper Functions

```
const PDFDocument = require('pdfkit');
const fs = require('fs');
const nodemailer = require('nodemailer');
const moment = require('moment');
const twilio = require('twilio');

const transporter = nodemailer.createTransport({
  service: 'gmail', // or your email provider
  auth: { user: 'your.email@gmail.com', pass: 'yourpassword' }
});

// Example Twilio client
const client = twilio('TWILIO_ACCOUNT_SID', 'TWILIO_AUTH_TOKEN');

function generatePDF(filename, title, data) {
  return new Promise((resolve, reject) => {
    const doc = new PDFDocument();
    const stream = fs.createWriteStream(filename);
    doc.pipe(stream);

    doc.fontSize(20).text(title, { align: 'center' });
    doc.moveDown();

    data.forEach(item => {
      doc.fontSize(12).text(JSON.stringify(item));
      doc.moveDown();
    });

    doc.end();
    stream.on('finish', () => resolve(filename));
    stream.on('error', reject);
  });
}

async function sendEmail(to, subject, text, attachmentPath) {
  const mailOptions = {
    from: 'your.email@gmail.com',
    to,
    subject,
    text,
    attachments: attachmentPath ? [{ path: attachmentPath }] : []
  };
  await transporter.sendMail(mailOptions);
}

async function sendSMS(to, message) {
  await client.messages.create({
    body: message,
    from: '+1234567890', // your Twilio number
    to
  });
}
```

11.2 Worker Processing

```
const { expiryReportQueue, birthdayQueue, discardQueue } = require('./queues');
const BloodBag = require('./models/BloodBag'); // Example Mongoose models
const Donor = require('./models/Donor');

// ------------ Case 1: Expiry Report ------------
expiryReportQueue.process('send-expiry-report', async (job) => {
  // 1. Get near-expiry blood (example: 5 days to expire)
  const today = moment().startOf('day');
  const nearExpiry = await BloodBag.find({ expiryDate: { $lte: moment(today).add(5, 'days')

  // 2. Generate PDF
  const pdfPath = `./reports/near_expiry_${Date.now()}.pdf`;
  await generatePDF(pdfPath, 'Near Expiry Blood Report', nearExpiry);

  // 3. Send Email
  await sendEmail('admin@bloodbank.com', 'Daily Near Expiry Report', 'Please find attached

  console.log('Expiry report sent successfully.');
});

// ------------ Case 2: Birthday SMS ------------
birthdayQueue.process('send-birthday-sms', async (job) => {
  const today = moment().format('MM-DD');
  const donors = await Donor.find({ birthday: { $regex: today } });

  for (const donor of donors) {
    await sendSMS(donor.phone, `Happy Birthday ${donor.name} from our Blood Bank!`);
  }

  console.log('Birthday SMS sent.');
});

// ------------ Case 3: Discard Expired Blood + Report ------------
discardQueue.process('discard-expired', async (job) => {
  const today = moment().startOf('day').toDate();
  const expiredBags = await BloodBag.find({ expiryDate: { $lt: today }, status: 'active' }

  // Update DB to mark discarded
  for (const bag of expiredBags) {
    bag.status = 'discarded';
    await bag.save();
  }

  // Generate PDF report
  const pdfPath = `./reports/discarded_${Date.now()}.pdf`;
  await generatePDF(pdfPath, 'Discarded Blood Report', expiredBags);

  // Send Email
  await sendEmail('admin@bloodbank.com', 'Daily Discarded Blood Report', 'See attached repo

  console.log('Discarded blood processed and report sent.');
});
```

12 Run Worker

Use PM2 to run continuously in background:

```
pm2 start worker.js --name bloodbank-worker
```

Bull + Redis ensures that jobs run automatically at scheduled time. Even if your server restarts, jobs are persistent in Redis.

13 Folder Structure Example

```
bloodbank-app/

  models/
     BloodBag.js
     Donor.js

  queues.js
  worker.js
  server.js
  reports/       # PDFs saved here
  package.json
```

14 Summary

1. Three separate Bull queues for each case.
2. Jobs scheduled using cron-like syntax.
3. Workers process jobs:
    - Case 1: Near-expiry → PDF → email
    - Case 2: Birthday → SMS
    - Case 3: Discard expired → DB updates → PDF → email
4. PDF generation via PDFKit.
5. Email via Nodemailer, SMS via Twilio.
6. Persistent, background execution with Redis + Bull.