```
// server.js
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const stripe = require('stripe')(process.env.STRIPE_SECRET_KEY);
const admin = require('firebase-admin');
const serviceAccount = require('./serviceAccountKey.json');

// Initialize Firebase Admin SDK (for Firestore database)
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)
});

const db = admin.firestore();

const app = express();

// Middleware
app.use(cors({ origin: true }));
app.use(bodyParser.json());
app.use(bodyParser.raw({ type: 'application/json' }));

// Routes
app.get('/auctions', async (req, res) => {
  try {
    const auctionsSnapshot = await db.collection('auctions')
      .where('endTime', '>', new Date())
      .orderBy('endTime', 'asc')
      .get();

    const auctions = [];
    auctionsSnapshot.forEach(doc => {
      auctions.push({
        id: doc.id,
        ...doc.data(),
        endTime: doc.data().endTime.toDate().toISOString()
      });
    });

    res.status(200).json(auctions);
  } catch (error) {
    console.error('Error fetching auctions:', error);
    res.status(500).json({ error: error.message });
  }
});

app.get('/bids', async (req, res) => {
  try {
    const { userId } = req.query;

    if (!userId) {
      return res.status(400).json({ error: 'User ID is required' });
    }

    const bidsSnapshot = await db.collection('bids')
      .where('userId', '==', userId)
      .orderBy('createdAt', 'desc')
      .get();

    const bids = [];
    bidsSnapshot.forEach(doc => {
      bids.push({
        id: doc.id,
        ...doc.data(),
        createdAt: doc.data().createdAt.toDate().toISOString()
      });
    });

    res.status(200).json(bids);
  } catch (error) {
    console.error('Error fetching bids:', error);
    res.status(500).json({ error: error.message });
  }
});

app.post('/bids', async (req, res) => {
  try {
    const { auctionId, userId, amount } = req.body;

    if (!auctionId || !userId || !amount) {
      return res.status(400).json({ error: 'Missing required fields' });
    }

    // Check if auction exists and still active
    const auctionDoc = await db.collection('auctions').doc(auctionId).get();

    if (!auctionDoc.exists) {
      return res.status(404).json({ error: 'Auction not found' });
    }

    const auction = auctionDoc.data();
    const endTime = auction.endTime.toDate();

    if (endTime < new Date()) {
      return res.status(400).json({ error: 'Auction has ended' });
    }
    // Check if bid amount is valid
    if (amount < auction.startingPrice) {
      return res.status(400).json({
        error: 'Bid amount must be higher than starting price'
      });
    }

    // Check if there's a higher bid
    const highestBidSnapshot = await db.collection('bids')
      .where('auctionId', '==', auctionId)
      .orderBy('amount', 'desc')
      .limit(1)
      .get();

    if (!highestBidSnapshot.empty) {
      const highestBid = highestBidSnapshot.docs[0].data();
      if (amount <= highestBid.amount) {
        return res.status(400).json({
          error: 'Bid amount must be higher than current highest bid'
        });
      }
    }

    // Create new bid
    const newBid = {
      auctionId,
      userId,
      amount,
      status: 'pending',
      createdAt: admin.firestore.FieldValue.serverTimestamp()
    };

    const bidRef = await db.collection('bids').add(newBid);

    // Get the created bid with proper timestamp
    const createdBid = await bidRef.get();

    res.status(201).json({
      id: bidRef.id,
      ...createdBid.data(),
      createdAt: createdBid.data().createdAt.toDate().toISOString()
    });
  } catch (error) {
    console.error('Error creating bid:', error);
    res.status(500).json({ error: error.message });
  }
});
```

```
app.patch('/bids/:bidId', async (req, res) => {
  try {
    const { bidId } = req.params;
    const { status } = req.body;

    if (!status) {
      return res.status(400).json({ error: 'Status is required' });
    }

    const bidRef = db.collection('bids').doc(bidId);
    const bidDoc = await bidRef.get();

    if (!bidDoc.exists) {
      return res.status(404).json({ error: 'Bid not found' });
    }

    await bidRef.update({ status });

    const updatedBid = await bidRef.get();

    res.status(200).json({
      id: bidId,
      ...updatedBid.data(),
      createdAt: updatedBid.data().createdAt.toDate().toISOString()
    });
  } catch (error) {
    console.error('Error updating bid:', error);
    res.status(500).json({ error: error.message });
  }
});

app.post('/create-payment-intent', async (req, res) => {
  try {
    const { bidId } = req.body;

    if (!bidId) {
      return res.status(400).json({ error: 'Bid ID is required' });
    }

    // Get bid information
    const bidRef = db.collection('bids').doc(bidId);
    const bidDoc = await bidRef.get();

    if (!bidDoc.exists) {
      return res.status(404).json({ error: 'Bid not found' });
    }
    const bid = bidDoc.data();

    if (bid.status !== 'won') {
      return res.status(400).json({ error: 'Only winning bids can be paid' });
    }

    // Create payment intent
    const paymentIntent = await stripe.paymentIntents.create({
      amount: Math.round(bid.amount * 100), // Convert to cents
      currency: 'usd',
      metadata: { bidId },
    });

    res.status(200).json({
      clientSecret: paymentIntent.client_secret,
    });
  } catch (error) {
    console.error('Error creating payment intent:', error);
    res.status(500).json({ error: error.message });
  }
});
```