Gopalganj Science and Technology University,
Gopalgonj-8100

Hashanuz Zaman
Student ID:20CSE005
Md Jobaer Hossain
Student ID:20CSE028
Iftaker Siddique Tanveer
Student ID:20CSE036

Under Supervision of
Dr. Syful Islam
Assistant Professor

19 May, 2025

Department of Computer Science & Engineering

Gopalganj Science and Technology University

Statement of Originality

It is hereby declared that the contents of this project is original and any part of it has not been submitted elsewhere for the award of any degree.

_____ _____

Signature of the Supervisor Signature of the Candidate

Date: Date:

Dedicated to my parents & honorable teachers

Department of Computer Science & Engineering

Gopalgonj Science and Technology University

Abstract

Micro-Task and Earning Platform (MERN Stack)

This project presents a **role-based web platform** designed to connect Workers, Task-Creators, and Administrators for efficient micro-task management and monetization. Built on the **MERN stack** (MongoDB, Express.js, React.js, Node.js), the system enables:

- **Task Workers** to browse, complete, and submit tasks for rewards, with a built-in coin system (20 coins = $1) and withdrawal functionality.
- **Task-Creators** to post tasks, review submissions, and manage payments via Stripe integration.
- **Administrators** to oversee platform operations, resolve disputes, and manage user roles.

Key features include **JWT authentication with Google OAuth**, real-time notifications, responsive design, and secure payment processing. The architecture employs:

- **Microservices** for auth, tasks, payments, and notifications
- **Sharded MongoDB** for scalable data storage
- **Redis caching** for performance optimization
- **ImageBB** for media uploads

The platform addresses the growing demand for decentralized micro-task solutions while ensuring security, scalability, and user-friendly workflows. Hosted on **Firebase (frontend)** and **Render (backend)**, it demonstrates full-stack proficiency in modern web development.

Acknowledgement

In this very special moment, first and foremost I would like to express my heartiest gratitude to the almighty Allah for allowing me to accomplish this Project successfully. I am really thankful for the enormous blessings that the Almighty has bestowed upon me not only during my study period but also throughout my life.

In achieving this goal, I have gone through the interactions with and help from other people, and would like to extend my deepest appreciation to those who have contributed to this dissertation itself in an essential way. I would like to express my heartfelt thanks to all of you for being with me with immense support and care and to make this work success. I am pleased to Acknowledge **Dr. Sayful Islam**, sir for his invaluable guidance during the course of project work.

Hashanuz Zaman
Student ID:20CSE005
Md Jobaer Hossain
Student ID:20CSE028
Iftaker Siddique Tanveer
StudentID:20CSE036

Table of Contents

Contents

Gopalgonj-8100 1

Department of Computer Science & Engineering 1

Abstract 4

Acknowledgement 5

1. Introduction 7

Tripartite Role System: 8

Technical Innovation: 8

Architectural Robustness: 8

2. Objectives 8

Technical Objectives 8

3. System Requirements 9

1. Hardware Requirements 9

2. Software Requirements 10

3. Development Environment 10

4. Production Environment 10

5. Browser Compatibility 11

6. Network Requirements 11

7. Security Requirements 11

4. Database Schema 12

1. User Schema 12

5. Withdrawal Schema 16

5. Entity Relationship Diagram (ERD) 18

6. Features 19

Core Features 19

1. User Authentication & Authorization 19

2. Task Management 19

3. Payment System 19

4. Real-Time Features 19
Role-Specific Features 20
For Workers 20
For Task-Creators 20
For Admins 20
Technical Features 20
Security Features 20
Future Roadmap 21

Chapter 1

1. Introduction

The **Micro-Task and Earning Platform** is a full-stack web application designed to bridge the gap between task creators and freelance workers through a secure, scalable, and user-friendly interface. Built on the **MERN stack** (MongoDB, Express.js, React.js, Node.js), this platform addresses the growing demand for decentralized microtask solutions in today's gig economy.

Problem Statement

Traditional task-management systems often lack:

- Role-based workflows for workers, creators, and admins
- Transparent reward mechanisms
- Real-time progress tracking
- Secure payment integration

Solution Overview

Our platform introduces:

Tripartite Role System:

- *Workers* complete tasks for coins (convertible to cash)
- *Task-Creators* post and manage tasks with automated payments
- *Admins* oversee platform integrity

Technical Innovation:

- JWT authentication with Google OAuth
- Stripe-powered coin purchases/withdrawals
- Real-time notifications via WebSockets
- Responsive UI with mobile-first design

Architectural Robustness:

- Microservices for scalability
- Sharded MongoDB for performance
- Redis caching for frequent queries
-

Chapter 02

2. Objectives

Primary Goal

To develop a secure, scalable micro-task platform that enables seamless collaboration between workers and task creators through a transparent reward system.

Technical Objectives

1. Implement a **role-based access control** system with:

- Worker task submission & coin withdrawal
- Task-Creator task management & payment processing
- Admin oversight capabilities

2. Build a **full-stack MERN application** featuring:
   - JWT authentication with Google OAuth
   - Real-time notifications
   - Responsive mobile-first UI
3. Integrate **third-party services** for:
   - Payments (Stripe API)
   - Image uploads (ImageBB)
   - Performance monitoring
4. Ensure **system reliability** through:
   - Database sharding (MongoDB)
   - Redis caching
   - Automated backups

Chapter 3

3. System Requirements

1. Hardware Requirements

Component	Minimum Specs	Recommended Specs
Server	2 CPU Cores, 4GB RAM	4 CPU Cores, 8GB RAM

Component	Minimum Specs	Recommended Specs
Database	2 CPU Cores, 8GB RAM	4 CPU Cores, 16GB RAM
Client	Dual-core processor, 2GB RAM	Quad-core, 4GB RAM

2. Software Requirements

Backend:

- Node.js v16+
- Express.js v4.18+
- MongoDB v6.0+ (with sharding for production)
- Redis v7.0+ (for caching)

Frontend:

- React.js v18+
- TailwindCSS v3.3+
- Axios v1.3+
- React Router v6.8+

APIs/Services:

- Stripe API (Payment processing)
- ImageBB (Image uploads)
- Google OAuth (Authentication)

3. Development Environment

- VS Code (with ESLint/Prettier)
- Postman/Thunder Client (API testing)
- Git v2.35+ (Version control)
- Mermaid.js (For diagrams)

4. Production Environment

Hosting:

- **Frontend:** Firebase Hosting / Netlify

- **Backend:** Render / AWS EC2
- **Database:** MongoDB Atlas (M2+ cluster)

Scaling:

- Load balancer (NGINX)
- Auto-scaling group (for backend)
- Database read replicas

5. Browser Compatibility

Browser	Minimum Version
Chrome	v110+
Firefox	v110+
Edge	v110+
Safari	v15+

6. Network Requirements

- HTTPS (TLS 1.3)
- 10+ Mbps bandwidth (for admin portal/media uploads)
- WebSocket support (for real-time notifications)

7. Security Requirements

- JWT token expiration (24h)
- Rate limiting (100 requests/min per IP)
- Database encryption at rest
- Regular backups (daily snapshots)

Chapter 4

4. Database Schema

1. User Schema

javascript

Copy

Download

```
{
  _id: ObjectId,
  name: { type: String, required: true },
  email: {
    type: String,
    required: true,
    unique: true,
    match: /^[^\s@]+@[^\s@]+\.[^\s@]+$/
  },
  password: {
    type: String,
    required: true,
    minlength: 8
  },
  role: {
    type: String,
    enum: ["worker", "task-creator", "admin"],
    default: "worker"
  },
  photoURL: { type: String },
```

```
  coinBalance: {
    type: Number,
    default: function() {
      return this.role === "task-creator" ? 50 : 10;
    }
  },
  createdAt: { type: Date, default: Date.now },
  updatedAt: { type: Date, default: Date.now }
}
```

Indexes:

javascript

Copy

Download

```
db.users.createIndex({ email: 1 }, { unique: true });
db.users.createIndex({ role: 1 });
```

2. Task Schema

javascript

Copy

Download

```
{
  _id: ObjectId,
  title: {
    type: String,
    required: true,
    maxlength: 100
  },
  description: { type: String, required: true },
  imageURL: { type: String },
  quantity: {
```

```
    type: Number,
    required: true,
    min: 1
  },
  payableAmount: {
    type: Number,
    required: true,
    min: 1
  },
  completionDate: { type: Date, required: true },
  creatorId: {
    type: ObjectId,
    ref: "User",
    required: true
  },
  status: {
    type: String,
    enum: ["active", "expired", "completed"],
    default: "active"
  },
  createdAt: { type: Date, default: Date.now }
}
```

Indexes:

```
javascript
db.tasks.createIndex({ creatorId: 1 });
db.tasks.createIndex({ status: 1, completionDate: 1 });
```

3. Submission Schema

```
javascript
{
```

```
  _id: ObjectId,
  taskId: {
    type: ObjectId,
    ref: "Task",
    required: true
  },
  workerId: {
    type: ObjectId,
    ref: "User",
    required: true
  },
  submissionDetails: { type: String, required: true },
  status: {
    type: String,
    enum: ["pending", "approved", "rejected"],
    default: "pending"
  },
  reviewedAt: { type: Date },
  createdAt: { type: Date, default: Date.now }
}
db.submissions.createIndex({ taskId: 1, status: 1 });
db.submissions.createIndex({ workerId: 1 });
```

4. Payment/Coin Purchase Schema

```
{
  _id: ObjectId,
  userId: {
    type: ObjectId,
    ref: "User",
    required: true
```

```
  },
  amount: { type: Number, required: true },
  coinsAdded: { type: Number, required: true },
  transactionId: { type: String, required: true }, // Stripe charge ID
  paymentMethod: { type: String, enum: ["stripe"] },
  status: {
    type: String,
    enum: ["pending", "completed", "failed"],
    default: "pending"
  },
  createdAt: { type: Date, default: Date.now }
}
```

5. Withdrawal Schema

javascript

Copy

Download

```
{
  _id: ObjectId,
  workerId: {
    type: ObjectId,
    ref: "User",
    required: true
  },
  coins: {
    type: Number,
    required: true,
    min: 20 // Minimum withdrawal = 20 coins ($1)
  },
  paymentSystem: {
```

```
    type: String,
    enum: ["bkash", "rocket", "nagad"],
    required: true
  },
  accountNumber: { type: String, required: true },
  status: {
    type: String,
    enum: ["pending", "processed", "rejected"],
    default: "pending"
  },
  processedAt: { type: Date },
  createdAt: { type: Date, default: Date.now }
}
```

6. Notification Schema

```
{
  _id: ObjectId,
  userId: {
    type: ObjectId,
    ref: "User",
    required: true
  },
  message: { type: String, required: true },
  type: {
    type: String,
    enum: ["task", "payment", "system"],
    required: true
  },
  isRead: { type: Boolean, default: false },
  relatedEntity: { // For deep linking
```

```
    type: String,
    enum: ["task", "submission", "withdrawal"]
  },
  entityId: { type: ObjectId }, // ID of related entity
  createdAt: { type: Date, default: Date.now }
}
```

Chapter 5

5. Entity Relationship Diagram (ERD)

Collection	Relations	Description
Users	1:N with Tasks	Creator owns tasks
Tasks	1:N with Submissions	Task has many submissions
Users	1:N with Submissions	Worker makes submissions
Users	1:N with Payments	User buys coins
Users	1:N with Withdrawals	Worker requests withdrawals

Chapter 6

6. Features

The Book Store Website is designed with specific features for both **Users** and **Administrators**.

Core Features

1. User Authentication & Authorization

- 🔐 **JWT-based login/logout with token refresh**
- 🔑 **Google OAuth integration (auto-register as Worker)**
- 🛡️ **Role-based access control (Worker/Task-Creator/Admin)**
- 🔄 **Persistent sessions (no redirect on page reload)**

2. Task Management

- 📝 **Task creation wizard (title, description, reward, deadline)**
- 🖼️ **Image uploads via ImageBB API**
- 🔍 **Task browsing with filters (status, reward, deadline)**
- 📊 **Dashboard analytics (tasks posted/completed)**

3. Payment System

- 💰 **Coin economy (1 USD = 20 coins)**
- 💳 **Stripe integration for coin purchases**
- 📤 **Withdrawal requests (Bkash/Rocket/Nagad)**
- 📜 **Transaction history for all users**

4. Real-Time Features

- 🔔 **WebSocket notifications for:**
  - **Task approvals/rejections**
  - **Payment confirmations**
  - **Admin actions**
- ⏳ **Countdown timers for task deadlines**

Role-Specific Features

For Workers

- 🏆 **Task submission system (text/upload)**
- 💸 **Coin balance tracker with withdrawal options**
- 📈 **Earnings dashboard (approved/rejected tasks)**
- 🏅 **Leaderboard (top earners)**

For Task-Creators

- ⚙️ **Task CRUD operations (Create/Read/Update/Delete)**
- 🔎 **Submission review panel (approve/reject with feedback)**
- 📉 **Coin management (auto-deduct on task creation)**
- 🗓️ **Task calendar view (deadline tracking)**

For Admins

- 👥 **User management (role changes, bans)**
- ⚠️ **Report resolution center (flagged content)**
- 💰 **Withdrawal approval system**
- 🗑️ **Content moderation (task/user deletion)**

Technical Features

- 🚀 **MERN stack architecture (MongoDB, Express, React, Node)**
- 🧩 **Microservices design (auth, tasks, payments)**
- ⚡ **Redis caching for frequent queries**
- 📱 **Mobile-first responsive UI (TailwindCSS)**
- 🛠️ **CI/CD pipeline (GitHub Actions)**

Security Features

- 🔒 **Environment variables for sensitive data**
- 🚫 **Rate limiting (100 requests/min)**

- 📝 **Input validation on all forms**
- 🕵️‍♂️ **Activity logging for admin actions**

Future Roadmap

- 📧 **Email notifications (SendGrid/Nodemailer)**
- 🌍 **Multi-language support**
- ⚖️ **Dispute resolution system**
- 📊 **Advanced analytics (user engagement)**

Feature Highlights Table

Feature	Worker	Task-Creator	Admin
Task Submission	✅	❌	❌
Coin Purchase	❌	✅	❌
User Banning	❌	❌	✅
Task Approval	❌	✅	✅