PeerConnect - Student Collaboration Platform

Complete Project Documentation

---

1. PROJECT OVERVIEW

1.1 What is PeerConnect?

PeerConnect is a cloud-based web application designed to connect students who want to collaborate on academic projects. It serves as a centralized platform where students can discover projects based on their skills, find team members with complementary abilities, and manage their collaborative work effectively.

1.2 Problem Statement

Students often face challenges in finding suitable projects that match their skills and interests. Similarly, students with project ideas struggle to find the right team members who possess the required technical expertise. This mismatch leads to:

- Missed collaboration opportunities
- Incomplete project teams
- Time wasted searching for teammates
- Difficulty in finding projects aligned with learning goals

1.3 Solution

PeerConnect solves these problems by providing:

- **Skill-based matching**: Students can browse projects that match their technical abilities
- **Project discovery**: Easy search and filter mechanisms to find relevant projects
- **Team management**: Tools to manage project teams, track members, and monitor progress
- **Collaborative workspace**: Centralized location for project documents and meeting notes
- **Invitation system**: Project owners can invite students based on their skill profiles

---

2. KEY FEATURES & FUNCTIONALITIES

2.1 User Authentication & Profile Management

- **Secure Registration**: Students register using university email addresses
- **JWT-based Authentication**: Token-based security for API access
- **User Profiles**: Students create profiles listing their technical skills
- **Skill Management**:
  - Add multiple skills (e.g., Java, Python, React)
  - Set proficiency levels (Beginner, Intermediate, Advanced, Expert)
  - Update and track skill progression over time
- **Profile Analytics**: View skills summary and proficiency distribution

2.2 Project Management

Project Creation:

- Create new projects with detailed information

- Define project goals and objectives
- Set maximum team size (e.g., 6 members)
- Specify required skills for team members
- Choose technology stack
- Set project status (RECRUITING, IN_PROGRESS, COMPLETED)

Project Discovery:

- Browse all available projects
- Search by title, description, or skills
- Filter by:
  - Project status
  - Required skills
  - Team availability
  - Technology stack
- View detailed project information

Project Details Include:

- Project title and description
- Problem statement
- Goals and objectives
- Current team size vs maximum team size (e.g., 3/6 members)
- List of completed objectives
- Required skills for joining
- Technology stack being used
- Project timeline (start date, end date)
- Project lead information

2.3 Team Collaboration

Join Requests:

- Students can request to join projects
- Project leads receive notifications
- Accept or reject join requests
- View requester's skill profile before decision

Invitation System:

- Project members can invite other students
- Browse student profiles and skills
- Send personalized invitations
- Track invitation status (pending, accepted, declined)

Team Management:

- View all team members
- See member roles (Project Lead, Member)
- Track member contributions
- Remove members if needed
- Monitor team size limits

2.4 Document Management

- **Upload Project Documents**: Store project-related files
- **Organized Storage**: Documents linked to specific projects
- **Access Control**: Only team members can view project documents
- **File Types Supported**: PDFs, Word documents, spreadsheets, presentations
- **Version Tracking**: Keep track of document versions

2.5 Meeting Notes

- **Create Meeting Notes**: Document team discussions and decisions
- **Timestamp Tracking**: Record when meetings occurred
- **Action Items**: Track tasks assigned during meetings
- **Searchable History**: Find past meeting notes easily
- **Team Access**: All team members can view and add notes

2.6 Skills Discovery

- **Browse All Skills**: Explore what technical skills students possess
- **Skill-based Search**: Find students with specific skills
- **Proficiency Filtering**: Search by skill level
- **Skill Analytics**: View top skills in the platform
- **Skill Categories**: Skills organized by categories (Programming, Design, Database, etc.)

2.7 Dashboard & Quick Actions

- **Personalized Dashboard**: Welcome screen with user-specific information
- **Quick Actions**:
  - Start New Project
  - Browse Projects
  - Discover Skills
- **Activity Summary**:
  - My Projects count
  - Connections (students in network)
  - Skills verified
- **Project Overview**: See all active collaborations
- **Notifications**: Alerts for join requests, invitations, updates

2.8 Project Workflow

Typical User Journey:

1. **Registration & Profile Setup**
   - Student registers with university email
   - Adds skills (Java, Python, React, etc.)
   - Sets proficiency levels
   - Completes profile
2. **Discovering Projects**
   - Browse available projects
   - Filter by skills and interests
   - View project details
   - Check team size and requirements
3. **Joining a Project**
   - Request to join interesting project
   - Wait for project lead approval
   - Receive notification on acceptance
   - Get access to project workspace
4. **Collaboration**
   - Upload and share documents
   - Create meeting notes
   - Invite additional team members
   - Track project progress
   - Update objectives
5. **Project Completion**
   - Mark objectives as complete
   - Update project status

    - Add final documents
    - Close project

---

3. TECHNICAL ARCHITECTURE

3.1 Architecture Type

Monolithic Architecture: The application is built as a single, unified unit where frontend, backend, and business logic are tightly integrated and deployed together.

Benefits for PeerConnect:

- Simpler to develop initially
- Easier to test as single unit
- Straightforward deployment process
- Lower complexity for small team
- All components in one codebase

Future Consideration: As the platform grows to serve multiple universities, migration to microservices architecture may be beneficial for better scalability.

3.2 Technology Stack

Frontend Technologies:

- **React**: JavaScript library for building user interface
- **Component-based Architecture**: Reusable UI components
- **Responsive Design**: Works on desktop, tablet, and mobile
- **Modern UI**: Clean, intuitive interface
- **Real-time Updates**: Dynamic content loading

Backend Technologies:

- **Spring Boot**: Java-based backend framework
- **RESTful APIs**: Standard HTTP API design
- **JWT Authentication**: Secure token-based authentication
- **Business Logic Layer**: Handles all application logic
- **Data Validation**: Ensures data integrity
- **Error Handling**: Robust exception management

Database Technologies:

- **MySQL**: Primary relational database for structured data
- **TiDB Cloud**: Cloud-native distributed SQL database
- **Tables Include**:
    - Users (student information, authentication)
    - Projects (project details, status)
    - Skills (skill information, proficiency)
    - Project_Members (team membership)
    - Documents (file metadata)
    - Meeting_Notes (collaboration notes)
    - Invitations (join requests and invites)

API Documentation:

- **Swagger**: Interactive API documentation
- **Endpoint Testing**: Test APIs directly from documentation
- **Request/Response Examples**: Clear API usage examples

- **Authentication Testing**: Test JWT token flows

3.3 Cloud Infrastructure

Cloud Platform:

- **Amazon Web Services (AWS)**: Primary cloud provider
- **Region**: Deployed in AWS region closest to users (e.g., Mumbai for India)
- **High Availability**: Cloud infrastructure ensures uptime
- **Scalability**: Can scale resources based on demand

Compute Resources:

- **AWS EC2 (Elastic Compute Cloud)**: Virtual servers hosting the application
- **Instance Types**:
  - t3.medium/large for frontend
  - t3.large/m5.large for backend
  - r5.large for database (if self-hosted)
- **On-Demand Scaling**: Add more instances during peak usage

Database Hosting:

- **TiDB Cloud**: Managed database service
- **Benefits**:
  - Automatic backups
  - High availability
  - Horizontal scalability
  - MySQL compatibility
  - Reduced operational overhead

Storage:

- **Document Storage**: Cloud storage for project documents
- **Persistent Storage**: Data retained even if instances restart
- **Backup Strategy**: Regular automated backups

---

4. CLOUD COMPUTING INTEGRATION

4.1 Deployed Cloud Concepts

Virtualization:

- Application runs on virtual machines (AWS EC2 instances)
- VMs provide isolated, dedicated computing resources
- Flexibility to choose different instance types based on workload

Data Storage:

- Cloud-based database (TiDB Cloud)
- Eliminates need for physical database servers
- Automatic data replication for redundancy
- Scalable storage capacity

Pay-as-you-go Model:

- Only pay for resources actually used
- Can scale down during low-usage periods (semester breaks)
- Scale up during high-demand periods (registration, project deadlines)

- Cost-effective compared to maintaining physical infrastructure

4.2 Potential Cloud Enhancements

Load Balancing:

- Distribute traffic across multiple EC2 instances
- Ensure no single server becomes bottleneck
- Improve response times during peak usage
- Automatic failover if one instance fails

Auto-Scaling:

- Automatically add EC2 instances when CPU usage exceeds threshold (e.g., 70%)
- Scale down when demand decreases
- Handle sudden traffic spikes (e.g., project submission deadlines)
- Optimize costs by running minimum necessary resources

Containerization with Docker:

- Package application components in containers
- Faster deployment and updates
- Consistent environments across development, testing, production
- Better resource utilization
- Easier to scale individual components

Container Orchestration with Kubernetes:

- Manage multiple containers across multiple servers
- Automatic container restart on failure
- Self-healing infrastructure
- Rolling updates with zero downtime
- Advanced load balancing and service discovery

Multitenancy:

- Serve multiple universities on same infrastructure
- Data isolation between universities
- Shared resources, reduced costs per university
- Centralized maintenance and updates
- Scalable to hundreds of universities

Content Delivery Network (CDN):

- Cache static content (images, JavaScript, CSS) closer to users
- Faster page load times
- Reduced load on main servers
- Better performance for geographically distributed users

Monitoring & Analytics:

- Cloud monitoring tools to track performance
- Real-time alerts for issues
- Usage analytics for optimization
- Cost tracking and optimization recommendations

---

5. CLOUD COMPUTING CONCEPTS APPLIED

5.1 Instances (Virtual Machines)

Definition: Virtualized computers running on cloud infrastructure

PeerConnect Application:

- Frontend deployed on separate instance
- Backend API deployed on another instance
- Can start, stop, or resize instances on-demand
- Pay only for running time
- Customize CPU, RAM, storage based on needs

Benefits:

- No physical hardware to maintain
- Quick provisioning (minutes, not weeks)
- Easy to upgrade/downgrade resources
- Can replicate instances for redundancy

5.2 Load Balancing

Definition: Distributing incoming requests across multiple servers

PeerConnect Scenario: During registration week, 5000 students access platform simultaneously:

- **Without Load Balancer**: Single server overwhelmed, slow response
- **With Load Balancer**: Traffic distributed across 5 servers, fast response

Load Balancing Algorithms:

- **Least Connections**: Routes to server with fewest active users (best for PeerConnect)
- **Round Robin**: Distributes requests equally
- **IP Hash**: Same student always connects to same server

Benefits:

- Improved performance during peak times
- High availability (if one server fails, others continue)
- Seamless user experience
- Better resource utilization

5.3 Virtualization

Definition: Creating virtual versions of computing resources

PeerConnect Benefits:

- Run multiple isolated environments on single physical server
- Development, testing, staging, and production on same hardware
- Cost savings (1 server instead of 4)
- Quick creation of new environments
- Easy backup and restore

Types Used:

- **Hardware Virtualization**: EC2 instances are virtual machines
- **Type 1 Hypervisor**: AWS uses KVM/Xen for better performance

- **OS Virtualization**: Potential Docker containerization

5.4 Task Scheduling

Definition: Algorithms to assign tasks to computing resources efficiently

PeerConnect Applications:

- **Background Jobs**:
  - Generate project recommendations (nightly batch)
  - Send email notifications (queued processing)
  - Clean up old data (scheduled maintenance)
  - Generate analytics reports (weekly)

Scheduling Algorithms:

- **FCFS**: Process join requests in order received
- **SJF**: Quick notifications before heavy reports
- **Min-Min**: Optimize batch processing overnight
- **Priority Scheduling**: User requests prioritized over background jobs

5.5 Containers (Potential Implementation)

Definition: Lightweight, portable packages containing application and dependencies

PeerConnect Containerization:

- **Frontend Container**: React app + Nginx
- **Backend Container**: Spring Boot + Java runtime
- **Database Container**: MySQL with configurations

Benefits:

- Consistent environments (no "works on my machine" issues)
- Faster deployment (containers start in seconds)
- Better resource efficiency (lower memory footprint)
- Easier scaling (spin up multiple containers quickly)
- Simplified CI/CD pipeline

5.6 Multitenancy (Future Enhancement)

Definition: Single application instance serving multiple organizations

PeerConnect Multi-University Deployment:

- Woxsen University students use woxsen.peerconnect.com
- MIT students use mit.peerconnect.com
- Stanford students use stanford.peerconnect.com
- All use same application and infrastructure

Implementation:

- University ID in every database table
- Subdomain routing to identify university
- Data isolation through query filters
- Custom branding per university

Benefits:

- 10 universities share infrastructure costs

- Single codebase, easier maintenance
- Updates deployed once, all universities benefit
- Optimal resource utilization across time zones

---

6. SYSTEM CAPABILITIES

6.1 Scalability

Current State:

- Handles 500 Woxsen University students
- Single university deployment

Future Scalability:

- **Horizontal Scaling**: Add more EC2 instances for increased load
- **Vertical Scaling**: Upgrade to larger instance types
- **Database Scaling**: TiDB Cloud handles distributed scaling
- **Multi-University**: Extend to 10-50 universities (5,000-25,000 students)

6.2 Performance

Response Times:

- Project search: < 500ms
- User authentication: < 200ms
- Document upload: Depends on file size
- Page loads: < 2 seconds

Optimization Strategies:

- Database indexing on frequently queried fields
- Caching for static content
- Asynchronous processing for heavy operations
- Load balancing for concurrent users

6.3 Reliability

Data Protection:

- Automated database backups (daily)
- JWT token expiration for security
- Input validation prevents malicious data
- Error handling ensures graceful failures

High Availability:

- Cloud infrastructure provides 99.9% uptime
- Potential for multi-instance deployment
- Database replication for redundancy

6.4 Security

Authentication:

- JWT (JSON Web Tokens) for secure API access
- Token-based authentication (no session storage)

- Password hashing (not stored in plain text)
- Email-based account verification

Authorization:

- Role-based access control (Project Lead, Member)
- Users can only access their projects
- Document access restricted to team members
- University-based data isolation (future multitenancy)

Data Privacy:

- Student information kept confidential
- Project data visible only to team members
- Secure communication (HTTPS)
- Compliance with data protection standards

---

7. USER WORKFLOW EXAMPLES

7.1 Example: Student Finding a Project

Step 1: Login

- Student visits peerconnect.com
- Enters email and password
- System validates and issues JWT token

Step 2: Browse Projects

- Student clicks "Browse Projects"
- Sees list of available projects with status "RECRUITING"
- Views: Title, description, team size (e.g., 3/6), required skills

Step 3: Filter by Skills

- Student has skills: Java, React, MySQL
- Applies skill filter
- System shows only projects requiring these skills

Step 4: View Project Details

- Student clicks on "E-Commerce Platform" project
- Sees full details: goals, tech stack, team members, timeline
- Checks if skills match requirements

Step 5: Request to Join

- Student clicks "Request to Join"
- Writes message to project lead
- System sends notification to project lead

Step 6: Acceptance

- Project lead reviews student's skill profile
- Approves join request
- Student receives notification
- Team size updates: 4/6

Step 7: Collaboration Begins

- Student accesses project workspace
- Views project documents
- Adds meeting notes from first team meeting
- Starts working on assigned tasks

7.2 Example: Creating a New Project

Step 1: Create Project

- Student has AI chatbot idea
- Clicks "Start New Project"
- Fills form:
    - Title: "AI-Powered Study Assistant"
    - Description: "Chatbot to help students with homework"
    - Max Team Size: 4
    - Status: RECRUITING

Step 2: Define Requirements

- Sets objectives:
    - Build NLP model
    - Create web interface
    - Deploy on cloud
    - Integrate with university system
- Specifies required skills:
    - Python
    - TensorFlow
    - React
    - AWS

Step 3: Invite Team Members

- Browses student profiles
- Finds student with Python & TensorFlow skills
- Sends invitation
- Student accepts

Step 4: Project Management

- Uploads project proposal document
- Creates meeting notes for kickoff
- Team starts development
- Updates objectives as completed

---

8. DEPLOYMENT SCENARIOS

8.1 Current Deployment (Assumption)

Single University - Basic Setup:

- 1 AWS EC2 instance (m5.large)
- React frontend and Spring Boot backend on same instance
- TiDB Cloud database (managed service)
- Serves 500 Woxsen University students
- Cost: ~$150-200/month

8.2 Improved Deployment

Load Balanced Setup:

- 3 EC2 instances behind Application Load Balancer
- Instance 1: Frontend (t3.medium)
- Instance 2-3: Backend (m5.large)
- TiDB Cloud database
- Auto-scaling: 2-5 instances based on CPU usage
- Cost: ~$400-600/month (scales with usage)

8.3 Containerized Deployment

Docker + Kubernetes:

- Kubernetes cluster with 3 worker nodes
- Frontend: 2 replicas (containers)
- Backend: 3 replicas (containers)
- Database: 1 replica with persistent storage
- Auto-scaling: 2-10 backend replicas
- Zero-downtime rolling updates
- Cost: ~$500-800/month

8.4 Multi-University Deployment

Enterprise Scale:

- 10 universities, 10,000 total students
- Kubernetes cluster across multiple availability zones
- Frontend: 5 replicas
- Backend: 10 replicas
- Database: Distributed TiDB cluster
- CDN for static content
- Multi-region deployment for global access
- Cost: ~$2,000-3,000/month (shared across universities)
- Cost per university: $200-300/month

---

9. TECHNOLOGY DECISIONS

9.1 Why React?

- **Component Reusability**: Build once, use everywhere
- **Fast Rendering**: Virtual DOM for optimal performance
- **Large Ecosystem**: Many libraries and tools available
- **Community Support**: Easy to find solutions
- **Modern UI**: Create responsive, interactive interfaces

9.2 Why Spring Boot?

- **Rapid Development**: Built-in features reduce coding time
- **Industry Standard**: Widely used in enterprise applications
- **Strong Security**: Built-in authentication and authorization
- **Easy Integration**: Works well with databases, APIs
- **RESTful APIs**: Standard web service architecture

9.3 Why MySQL + TiDB Cloud?

- **Relational Data**: Student, project, skill relationships
- **ACID Compliance**: Data consistency and integrity
- **SQL Familiarity**: Standard query language
- **TiDB Cloud Benefits**:
    - Horizontal scalability (handle growth)
    - High availability (automatic failover)
    - MySQL compatible (easy migration)
    - Managed service (less operational overhead)

9.4 Why AWS?

- **Market Leader**: Most mature cloud platform
- **Comprehensive Services**: Everything needed in one place
- **Reliability**: 99.99% uptime SLA
- **Global Infrastructure**: Deploy closer to users
- **Cost-Effective**: Pay-as-you-go pricing
- **Learning Opportunity**: Industry-relevant experience

9.5 Why Monolithic Architecture?

Current Phase:

- Simpler to develop with small team
- Easier deployment and testing
- Lower operational complexity
- Faster initial development

Future Consideration: When scaling to multiple universities, consider microservices:

- Independent scaling of components
- Better fault isolation
- Technology flexibility per service
- Easier team collaboration

---

10. SUCCESS METRICS

10.1 User Engagement

- Number of registered students
- Active projects count
- Join requests per week
- Document uploads per project
- Meeting notes created

10.2 Performance Metrics

- Average page load time
- API response time
- System uptime percentage
- Concurrent users handled

10.3 Business Metrics

- Student satisfaction ratings

- Project completion rate
- Team formation success rate
- Platform adoption across universities

---

11. FUTURE ENHANCEMENTS

11.1 Short-term (3-6 months)

- Real-time chat between team members
- Video meeting integration
- Mobile application (iOS/Android)
- Enhanced search with AI recommendations
- Email notifications for updates

11.2 Medium-term (6-12 months)

- Multi-university deployment
- Advanced analytics dashboard
- Project templates and best practices
- Integration with GitHub for code repositories
- Skill assessment and certification

11.3 Long-term (1-2 years)

- AI-powered project matching
- Blockchain for achievement verification
- Industry partnership for real-world projects
- Gamification with badges and leaderboards
- Global student collaboration across countries

---

12. CONCLUSION

PeerConnect successfully addresses the challenge of connecting students for collaborative projects by providing a comprehensive, cloud-based platform. Built using modern technologies (React, Spring Boot, MySQL) and deployed on AWS cloud infrastructure, the application demonstrates practical implementation of cloud computing concepts including virtualization, scalability, and distributed data storage.

The platform's architecture allows for future growth, with clear paths to implementing advanced cloud features like load balancing, containerization, and multitenancy. As the platform expands to serve multiple universities, these enhancements will ensure PeerConnect continues to deliver excellent performance and user experience while maintaining cost-effectiveness.

Key Achievements: ✓ Solves real student collaboration problem ✓ Modern, scalable technology stack ✓ Cloud-native architecture ✓ Practical application of cloud computing concepts ✓ Foundation for multi-university expansion ✓ Industry-relevant implementation experience

PeerConnect represents not just a functional application, but a learning platform for understanding cloud computing, distributed systems, and modern web application development.