🎓 **Project: Ticketing Gateway System**

🗓️ **Timeline: 4 Weeks**

👥 **Team Size: 3–5 Team Members**

🧱 **Tech Stack:**

- **Backend**: Java Spring Boot, Spring Security, Spring Data JPA, ActiveMQ, JavaMailSender
- **Frontend**: HTML/CSS, jQuery, AJAX
- **Database**: MySQL/Oracle
- **Other Tools**: Postman, Git/GitHub, iText PDF, cron scheduler

---

🔶 Week 1: Project Setup & Basic Architecture

Goals:

- Understand microservices architecture
- Set up base repositories and services
- Create core entities (Ticket, Employee, Role, TicketHistory)

Tasks:

- ☐ Initialize GitHub repo and project structure (monorepo or per microservice)
- ☐ Create Employee and Role entities (with @ManyToMany)
- ☐ Implement Spring Security and login form
- ☐ Set up database (MySQL) and connect using Spring JPA
- ☐ Build basic login/authentication flow
- ☐ Setup microservices:
  - Ticketing Gateway (UI/Frontend)
  - Ticket Microservice (CRUD + history)
  - Notification Microservice (basic skeleton)
- ☐ Develop simple frontend with login and dashboard per role

Deliverables:

- Authenticated login for USER, MANAGER, ADMIN
- Database tables created and connected

- GitHub repo with initial commits

🔶 Week 2: Ticket Workflow Implementation

Goals:

- Implement CRUD ticket operations
- Enable approval, rejection, resolution, and viewing of tickets

Tasks:

- ☐ Ticket creation from UI with file upload support
- ☐ Save ticket info via REST API to Ticket Microservice
- ☐ Implement approval/rejection by MANAGER
- ☐ Ticket resolution by ADMIN
- ☐ Reopen/close functionality
- ☐ Create endpoints for ticket history viewing

Deliverables:

- Fully working ticket lifecycle in database
- UI components for creating, viewing, and updating tickets
- REST APIs with basic test coverage (Postman or Swagger)

🔶 Week 3: Notifications, Automation, and Messaging

Goals:

- Implement automated emails and scheduled tasks
- Set up inter-microservice communication using ActiveMQ

Tasks:

- ☐ Send email on ticket creation (SimpleMailMessage)
- ☐ CRON job: Check tickets pending >7 days → notify manager
- ☐ Generate dynamic PDFs (iText) on ticket resolution

- ☐ Send PDF via MimeMessage email
- ☐ Implement ActiveMQ setup:
  - Producer in Notification Service
  - Consumer in Ticket Service
- ☐ Auto-close tickets after 5 days of unresolved "Resolved" status

Deliverables:

- ActiveMQ working with Ticket & Notification services
- Scheduled tasks running via cron
- Functional email sending with PDF attachment

🔶 Week 4: UI Polishing, Testing & Documentation

Goals:

- Make the UI clean and functional
- Write documentation and handle edge cases
- Conduct testing and code cleanup

Tasks:

- ☐ Frontend improvements (status colors, modals, alerts)
- ☐ Display ticket statuses visually
- ☐ Validate form inputs and handle exceptions
- ☐ Unit tests for Ticket and Notification services
- ☐ Write README with setup, architecture, and screenshots
- ☐ Prepare a project report or demo script

Deliverables:

- Fully working system with login, ticket lifecycle, and notifications
- Polished UI for users and admins
- GitHub repo with documentation and code

🧩 Bonus/Future Extensions (Optional if time permits):

- Add role-based dashboards with ticket stats
- Implement real-time notifications using WebSockets
- Use Docker to containerize services

📋 Final Submission Checklist:

Item	Status
Project hosted on GitHub	✅
README with setup instructions	✅
Functional microservices	✅
Ticket lifecycle end-to-end	✅
Email notifications working	✅
Message Queue integration	✅
PDF generation + email attachment	✅
Presentation or Demo video	✅

🧩 Domain Entities

1. Employee

- **Attributes:**
  - `id` (Long)
  - `name` (String)
  - `email` (String)
  - `password` (String, encrypted)
  - `roles` (List<Role>)
  - `department` (String)
  - `project` (String)
  - `managerId` (Long)

2. Role

- **Attributes:**
  - `id` (Long)
  - `name` (String) // USER, MANAGER, ADMIN

3. Ticket

- **Attributes:**

- `id` (Long)
- `title` (String)
- `description` (String)
- `createdBy` (Employee)
- `assignee` (Employee)
- `priority` (String) // LOW, MEDIUM, HIGH
- `status` (String) // OPEN, PENDING_APPROVAL, APPROVED, REJECTED, ASSIGNED, RESOLVED, CLOSED, REOPENED
- `creationDate` (Date)
- `category` (String)
- `fileAttachmentPath` (String)
- `history` (List<TicketHistory>)

4. TicketHistory

- **Attributes:**
  - `id` (Long)
  - `ticket` (Ticket)
  - `action` (String) // CREATED, APPROVED, REJECTED, ASSIGNED, RESOLVED, CLOSED, REOPENED
  - `actionBy` (Employee)
  - `actionDate` (Date)
  - `comments` (String)

🎯 Main Use Cases

1. **User Login/Authentication**
   - Secure login using Spring Security
   - Different dashboards for USER, MANAGER, ADMIN
2. **Raise Ticket**
   - User creates a ticket (title, description, priority, category, attachment)
   - Ticket is saved in Ticket Microservice with status = OPEN
3. **Ticket Approval**
   - Manager reviews newly raised ticket
   - Manager approves or rejects the ticket
   - Status updated to APPROVED or REJECTED
   - User is notified via email
4. **Ticket Assignment & Resolution**
   - Approved tickets are assigned to IT/Admin staff
   - Admin/IT resolves the ticket and updates status to RESOLVED
   - Resolution details added, dynamic PDF generated
5. **Ticket Closure/Reopen**

    - User receives resolution email (with PDF)
    - User can CLOSE ticket or REOPEN if not satisfied (within 7 days)
    - After 5 days in RESOLVED, auto-close is triggered if not acted upon
6. **View Ticket History**
    - Users and managers can view the history of any ticket
7. **Notifications**
    - Email notifications sent at key stages (creation, approval, rejection, resolution, auto-close)
    - Reminders if tickets are pending action for too long (via CRON)
8. **File Upload/Download**
    - Users attach files to tickets
    - IT/Admin can download attached files

🔄 Process Flow Diagram (Described Step-by-Step)

1. **User logs in**
    - Authenticated by Spring Security
    - Redirected to dashboard based on role
2. **User raises a ticket**
    - Fills form and uploads attachment
    - Ticket created with status = OPEN
    - Email sent to manager for approval
3. **Manager reviews ticket**
    - Approves → Ticket status = APPROVED
        - Email sent to IT/Admin for assignment
    - Rejects → Ticket status = REJECTED
        - Email sent to user with reason
4. **IT/Admin resolves ticket**
    - Ticket assigned to IT staff
    - IT staff adds resolution, status = RESOLVED
    - Email sent to user with resolution details and attached PDF
5. **User closes/reopens ticket**
    - User happy: closes ticket (status = CLOSED, history updated)
    - User unhappy: reopens ticket (status = REOPENED, goes back to IT/Admin for re-processing)
6. **Automated flows**
    - If ticket is PENDING for >7 days: CRON triggers reminder email to manager/admin
    - If ticket is RESOLVED but not closed for >5 days: JMS triggers auto-close, status updated, notification sent
7. **Ticket history maintained**

- All actions logged in TicketHistory, viewable by user/manager/admin

---

📝 Summary Table: Domain Entities & Main Use Cases

Entity	Main Use Cases Impacted
Employee	Login, ticket creation, approval, resolution
Role	Auth control, dashboard view
Ticket	All ticket lifecycle actions
TicketHistory	Logging, ticket tracking, reporting

⚡ Tips for Implementation

- Use UML diagrams for **Use Case Diagram** and **Class Diagram**.
- **Sequence diagrams** are helpful for visualizing the process flow if you want to go further.
- Focus on **roles and permissions** to avoid unauthorized access to tickets/actions.
- Modularize microservices for clear code separation.