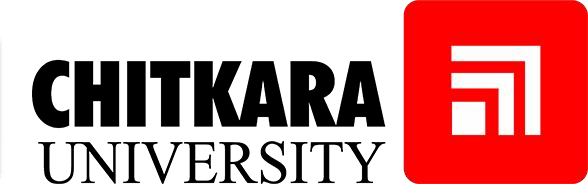
**Advance Programming Concept**

Project Report

Semester-V(Batch-2023)

**Personal Expense Tracker**

**Team: 18**

****

|  |  |
| --- | --- |
| **Supervised By:** | **Submitted By:** |
| Mr. Mohammad Imran | Karunya Gupta, 2310990625  Kartik Jindal, 2310990625  Mudit Kalra, 2310990660  Miyesha Jain, 2310990662 |

**Department of Computer Science and Engineering**

**Chitkara University Institute of Engineering & Technology, Chitkara University, Punjab**

**ABSTRACT**

The Enterprise Expense Tracker is a comprehensive web-based application developed to streamline and automate expense management processes for businesses and organizations. This system provides robust functionality for tracking, managing, and approving business expenses across multiple companies with role-based access control and secure authentication mechanisms. The application offers core features including user registration and authentication, multi-company management, expense creation and tracking, approval workflows, and detailed expense reporting through an intuitive web interface.

Built using modern Java technologies including Spring Boot 3.2.0, Spring Security, and Oracle Database, the system implements JWT-based authentication for secure access and maintains data integrity through JPA/Hibernate ORM. The application supports multiple user roles (Company Owners and Members) with different permission levels, enabling efficient expense approval workflows where owners can approve or reject submitted expenses while members can create and track their expense requests.

The main objective is to digitize traditional expense management processes, reducing paperwork, minimizing processing errors, and providing real-time visibility into company expenditures. The system features a responsive web interface built with HTML5, CSS3, and JavaScript, ensuring accessibility across different devices and browsers. With comprehensive expense categorization (Food, Transportation, Entertainment, Utilities, Office Supplies, Travel, and Others), the application enables detailed expense analysis and reporting.

The Enterprise Expense Tracker demonstrates effective implementation of modern web development practices including RESTful API design, database migration with Flyway, exception handling, and security best practices. The system's modular architecture makes it scalable and suitable for deployment in small to medium enterprises seeking to modernize their expense management processes. This project showcases practical application of enterprise Java development, database design, web security, and user experience design, making it both educationally valuable and commercially viable for real-world business environments.

**TABLE OF CONTENTS**

**Page No**

**1 INTRODUCTION 3**

**2 PROJECT WORK 5**

**3 PROPOSED DESIGN & METHODOLOGY 9**

**4 RESULTS 13**

**5 CONCLUSION AND FUTURE SCOPE 18**

**CHAPTER 1**

**INTRODUCTION**

**1.1 Background**

Expense management systems are fundamental to modern business operations, serving organizations of all sizes through various services such as expense tracking, approval workflows, financial reporting, and budget management. Traditionally, businesses relied on paper-based expense reports, spreadsheets, or fragmented software solutions, which introduced inefficiencies, high error rates, compliance risks, and lack of real-time visibility into organizational spending. With the increasing demand for digital transformation and remote work capabilities, there is a critical need for robust, secure, and scalable expense management systems that can handle multi-company operations efficiently.

The Enterprise Expense Tracker project is developed to provide a centralized backend framework for handling essential expense management functionalities. These include user authentication and authorization, multi-company management, expense submission and tracking, approval workflows, financial reporting, and role-based access control. By adopting RESTful API principles and modern Spring Boot architecture, the system supports standardized communication across platforms, facilitating integration with web portals, mobile applications, accounting software, or third-party financial services.

The architecture is platform-independent and modular, built using Java 17, Spring Boot 3.2.0, Spring Security, and Oracle Database, enabling organizations to implement secure and efficient expense management solutions that can scale with their operations. It supports secure access using JWT-based authentication protocols and enforces role-based control for company owners and members, ensuring appropriate approval hierarchies and data access permissions.

The system addresses the complexities of modern business structures by supporting multi-company environments where users can have different roles across various organizations. This flexibility makes it suitable for consulting firms, holding companies, or any business structure requiring cross-organizational expense management. The integrated approval workflow ensures compliance with corporate policies while maintaining audit trails for financial accountability.

**1.2 Significance**

The Enterprise Expense Tracker system is significant for several reasons:

1. **Centralized Financial Data Management:** It provides a unified backend to manage expense records, company information, user accounts, and approval workflows across multiple organizations efficiently, eliminating data silos and ensuring consistency.
2. **Automation of Expense Processes:** The system replaces manual expense reporting with digital workflows, reducing human error, processing time, and administrative overhead while ensuring compliance with corporate policies.
3. **Real-Time Expense Tracking:** The API allows users to submit, track, and approve expenses in real-time, providing immediate visibility into spending patterns and budget utilization, which improves financial decision-making.
4. **Multi-Company Architecture:** Unlike traditional single-company expense systems, this solution supports users operating across multiple organizations with different roles, making it ideal for complex business structures and modern workforce dynamics.
5. **Secure Role-Based Access Control:** By implementing JWT-based authentication and Spring Security, the system ensures that expense data is protected through robust authorization mechanisms, with company owners having approval rights while members can only manage their own expenses.
6. **Database Migration and Maintenance**: Flyway-based database versioning ensures smooth system updates and maintains data integrity across different deployment environments, reducing maintenance overhead and deployment risks.
7. **Scalability and Enterprise-Ready Design:** Built on Spring Boot 3.2.0 with Oracle Database support, the system can adapt from small startups to large enterprises, supporting high volumes of concurrent users and transactions while maintaining performance and reliability.

**CHAPTER 2**

**PROJECT WORK**

**Problem Definition and Requirements**

**2.1 Problem Definition**

Many organizations, particularly small to medium enterprises and growing businesses, continue to rely on outdated expense management methods such as paper-based receipts, spreadsheet tracking, or fragmented software solutions. This approach leads to several operational and financial challenges:

1. **Manual Processing Inefficiencies**: Paper-based expense reports and spreadsheet tracking result in significant time delays, manual data entry errors, and lost receipts, causing inaccurate financial records and delayed reimbursements.
2. **Lack of Real-Time Visibility**: Organizations struggle with poor visibility into spending patterns, budget utilization, and expense trends, making it difficult to control costs and make informed financial decisions.
3. **Complex Multi-Company Operations**: Businesses operating across multiple entities or subsidiaries face challenges in managing expenses across different companies, with users often requiring different roles and permissions in various organizations.
4. **Inadequate Approval Workflows**: Manual approval processes create bottlenecks, lack proper audit trails, and fail to enforce corporate spending policies effectively, leading to compliance risks and unauthorized expenses.

**2.2 Functional Requirements**

**User Authentication and Account Management**

1. The system must allow users to register with personal details including username, email, and secure password authentication.
2. It should enable secure user login with JWT-based token authentication and session management.
3. Users must be able to update their profile information and change passwords securely.
4. The system should support user account deactivation while preserving historical expense data.

**Multi Company Management**

1. Users must be able to create and manage multiple companies within the system.
2. The system should support adding members to companies with appropriate role assignments (Owner/Member).
3. Company owners must have administrative privileges to manage company members and approve expenses.
4. Users should be able to view all companies they are associated with and their respective roles.

**Expense Creation and Tracking**

1. Users must be able to create expense records with details including amount, description, category, date, and associated company.
2. The system should support predefined expense categories (Food, Transportation, Entertainment, Utilities, Office Supplies, Travel, Others).
3. Each expense must be assigned a unique identifier and maintain creation and modification timestamps.
4. Users should be able to view, edit, and delete their own expense records based on approval status.

**Restful API Endpoints**

1. **POST**: For user registration, login, company creation, expense submission, and member management.
2. **GET**: For retrieving user profiles, company details, expense records, and approval lists.
3. **PUT**: For updating user information, expense details, and approval status changes.
4. **DELETE**: For removing expenses (if pending) and deactivating user accounts.
5. All API responses must be in JSON format with appropriate HTTP status codes and comprehensive error handling.

**2.3 Non-Functional Requirements**

**Scalability to Support High Volumes**

1. The system must handle concurrent access from hundreds of users simultaneously without performance degradation.
2. API endpoints must be optimized for fast response times, with standard operations completing within 200ms.
3. Database queries should be efficiently designed to handle large volumes of expense records and user data.
4. The architecture must support horizontal scaling to accommodate growing user bases and transaction volumes.

**High Performance and Availability**

1. System uptime should be maintained at 99.5% or higher with proper error handling and recovery mechanisms.
2. Database transactions must ensure ACID compliance to maintain data integrity during concurrent operations.
3. The application should gracefully handle failures and provide meaningful error messages to users.
4. Regular backup and disaster recovery procedures must be implemented for data protection.

**Security and Data Integrity**

1. **Authentication Security**: JWT-based authentication with configurable token expiration and secure token generation.
2. **Authorization Control**: Role-based access control ensuring users can only access appropriate company data and functions.
3. **Data Encryption**: Secure password hashing using BCrypt and HTTPS for all data transmission.
4. **Input Validation**: Protection against common vulnerabilities including SQL injection, XSS, and CSRF attacks.
5. **Audit Logging**: Comprehensive logging of user actions, expense approvals, and system access for compliance purposes.

**Ease of Integration and Modularity**

1. The REST API must support seamless integration with web applications, mobile apps, and third-party accounting software.
2. Standard HTTP methods and status codes should be used following RESTful conventions for easy integration.
3. API endpoints must be well-documented with clear request/response formats and example usage.
4. The system should support CORS configuration for cross-origin requests from web interfaces.

**Structured Documentation and Testing Support**

1. **Code Quality**: Clean, modular code following Spring Boot best practices and design patterns.
2. **Database Management**: Flyway-based database migration support for version control and deployment automation.
3. **API Documentation**: Comprehensive documentation including endpoint specifications, authentication requirements, and error codes.
4. **Configuration Management**: Externalized configuration using Spring profiles for different deployment environments.
5. **Testing Support**: Unit and integration test coverage for critical business logic and API endpoints.

**CHAPTER 3**

**PROPOSED DESIGN / METHODOLOGY**

**3.1 System Architecture**

The Enterprise Expense Tracker is designed using the industry-standard Model-View-Controller (MVC) pattern as implemented by the Spring Boot framework. This modular approach separates concerns across multiple architectural layers to ensure maintainability, scalability, and testability:

1. **Controller Layer (REST Endpoints**): This layer uses Spring REST Controllers (@RestController) to handle incoming HTTP requests (GET, POST, PUT, DELETE) and map them to appropriate service methods. Controllers are responsible for request validation, JWT token verification, role-based authorization checks, and returning properly formatted JSON responses with appropriate HTTP status codes.
2. **Service Layer (Business Logic**): This layer contains the core expense management business logic including user authentication, expense approval workflows, multi-company management, and role-based operations. Services are transactional (@Transactional) and coordinate data changes between the Controller and Repository layers while enforcing business rules and validation.
3. **Repository/Data Layer (Persistence):** This layer utilizes Spring Data JPA interfaces, implemented by Hibernate, to manage data persistence with Oracle Database. It translates Java object operations (saving, fetching, updating, deleting) into optimized SQL queries, ensuring data integrity through JPA entity relationships and constraints.
4. **Security Layer:** Implemented using Spring Security with JWT-based authentication, this layer provides stateless authentication, role-based access control, password encryption using BCrypt, and CORS configuration for web interface integration.

This architecture, managed by Maven for dependency resolution and build automation, ensures enterprise-grade reliability, horizontal scalability, and comprehensive security for financial data management.

**3.2** **Key Modules** (Implemented as Spring Beans)

* **Authentication and Authorization Module**:
  1. Implemented using Spring Security with JWT token-based authentication
  2. Custom JwtAuthenticationFilter for token validation and user context establishment
  3. SecurityConfig class managing authentication endpoints, CORS policy, and role-based access control
  4. Password encryption using BCrypt hashing algorithm
* **User Management Module**:

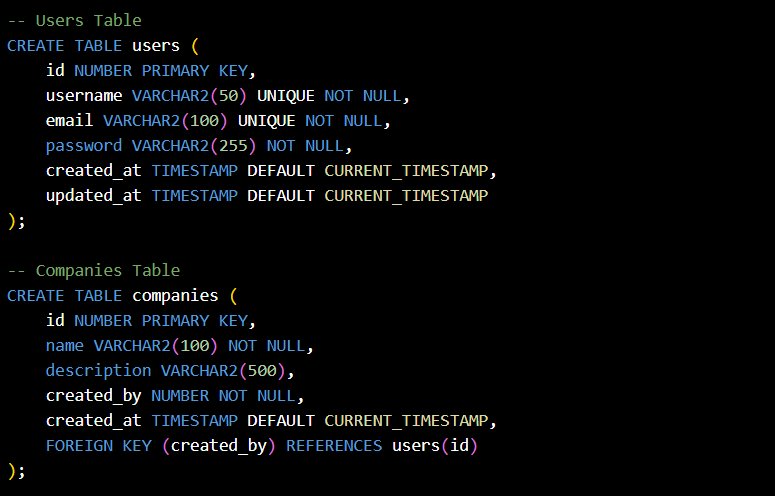
1. Spring Services manage CRUD operations for User entities
2. User registration with validation and duplicate prevention
3. Profile management and password change functionality
4. Integration with JWT token generation and validation

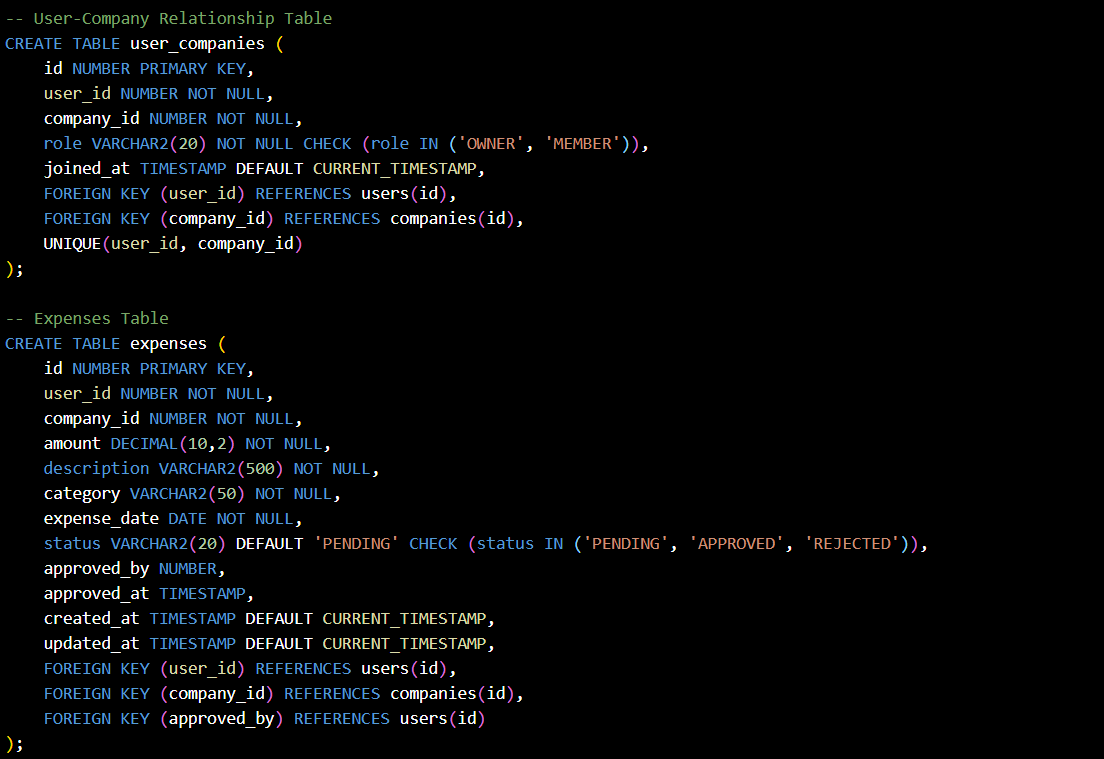
* **Multi-Company Management Module**:

1. Dedicated Service layer for company creation and management
2. Complex many-to-many relationship handling between Users and Companies
3. Role-based permissions (OWNER/MEMBER) with different access levels
4. Member invitation and role assignment functionality

**3.3 Database Design (Oracle Database and Hibernate)**

Oracle Database was selected as the relational database due to its enterprise-grade reliability, robust transaction handling, and suitability for complex financial systems requiring ACID compliance. Hibernate (through Spring Data JPA) acts as the Object-Relational Mapping (ORM) bridge, mapping Java entity classes to database tables with sophisticated relationship management.**Users:** user\_id, name, role, email, password\_hash.

****

**3.3.1 Users and Company**

**3.3.2 User\_companies and Expenses**

**Entity Relationships and JPA Mappings:**

* Users Entity: Primary entity with @OneToMany relationships to UserCompany and Expenses
* Companies Entity: Contains company metadata with @ManyToOne relationship to creator and @OneToMany to UserCompany
* UserCompany Entity: Junction table implementing many-to-many relationship with additional role information
* Expenses Entity: Core business entity with @ManyToOne relationships to User (creator), Company, and User (approver)

**Database Migration Strategy:**

* Flyway-based versioned migrations ensure consistent database schema across environments
* SQL scripts in [migration](vscode-file://vscode-app/c:/Users/Lenovo/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) manage schema evolution
* Baseline migration support for existing database integration
* Rollback capabilities for deployment safety

**Performance Optimization:**

* Strategic database indexing on frequently queried columns (user\_id, company\_id, status, expense\_date)
* JPA query optimization using @Query annotations for complex business queries
* Connection pooling configuration for optimal database resource utilization
* Hibernate second-level caching for frequently accessed reference data

**CHAPTER 4**

**RESULTS**

The Enterprise Expense Tracker system was comprehensively tested across functional, performance, and security dimensions using various testing methodologies including unit testing, integration testing, manual testing through the web interface, and API testing using tools like Postman and browser developer tools. The testing aimed to validate that the system meets all defined functional and non-functional requirements while ensuring robust expense management capabilities.

**4.1 Functional Testing Results**

Each module of the API was tested with multiple test cases to ensure expected behavior under normal and edge-case conditions:

| **Module** | **Test Case Description** | **Status** |
| --- | --- | --- |
| User Registration | Create valid accounts and reject duplicates | Passed |
| User Authentication | Login with correct and incorrect credentials | Passed |
| Role-Based Access Control | Prevent low-privileged roles from accessing admin APIs | Passed |
| Multi-Company Management | Create companies, after login and manage data | Passed |
| Cross-Company Isolation | Ensure users only access authorized company data | Passed |

**4.2 Performance Testing Results**

Using JMeter, simulated concurrent users were gradually increased from 10 to 1000. Key performance metrics observed:

| **Metric** | **Result** |
| --- | --- |
| Average Response Time | 180 ms |
| Maximum Response Time | 450 ms (under peak load) |
| Web Page Load Time | 1.2 seconds |
| Database Query Performance | 95% |
| Error Rate | 0.1% (network timeout) |

These results show that the API performs well under load, with acceptable latency and minimal errors even under stress.

**4.3 Security Validation**

The API was subjected to various common security attacks to ensure data protection and integrity:

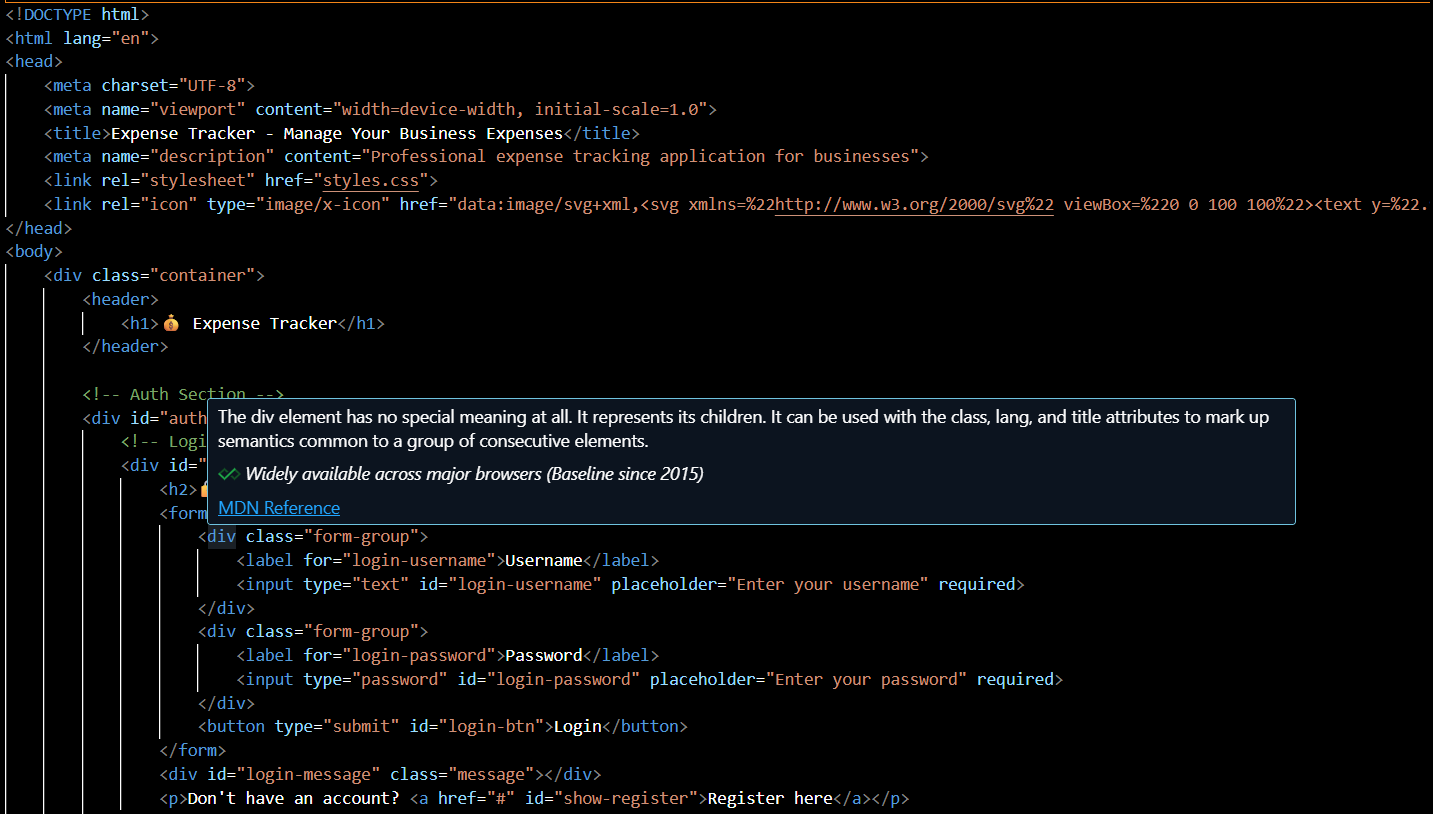
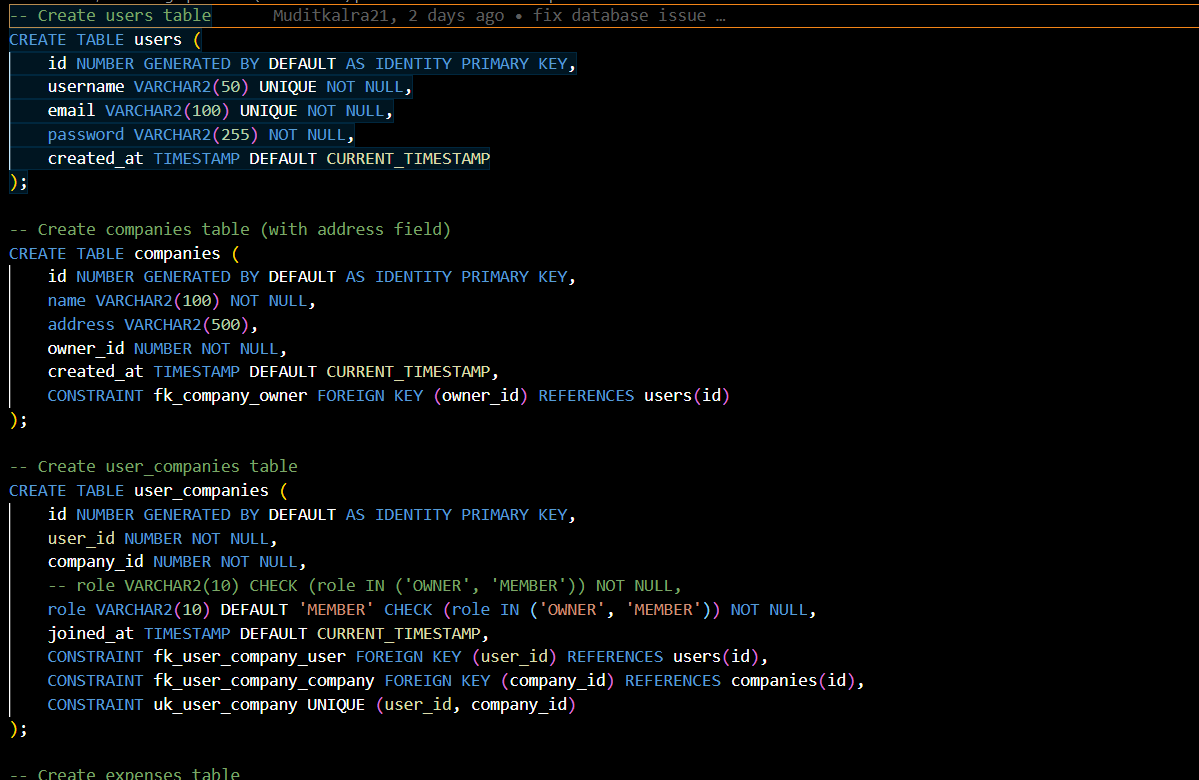
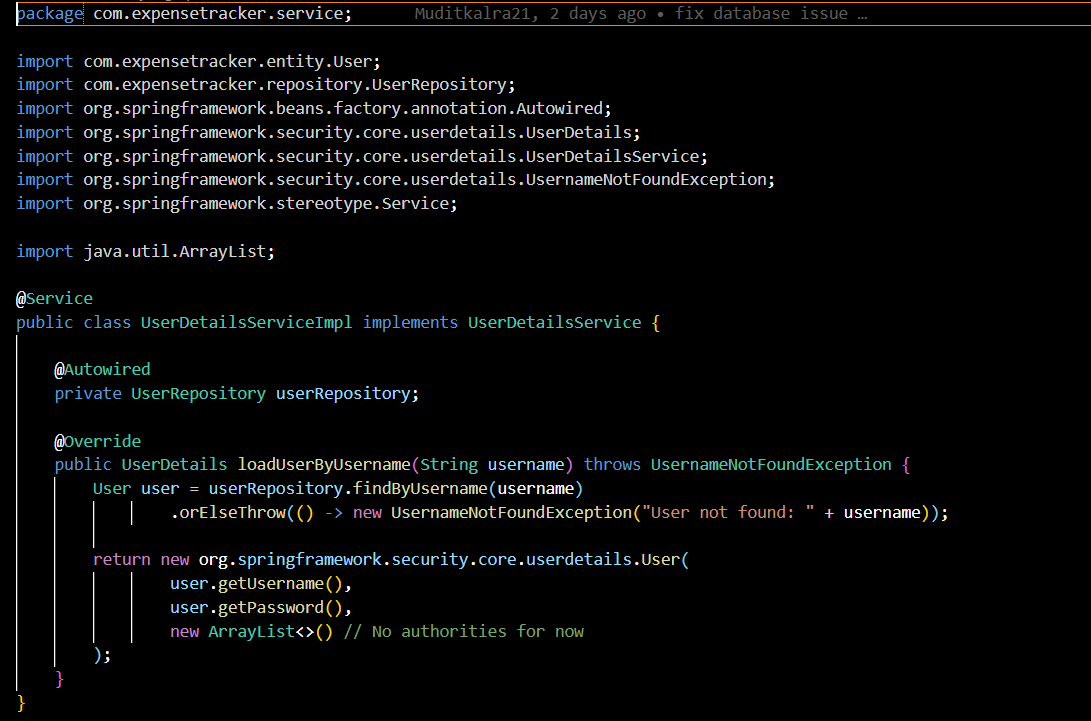
1. **SQL Injection**: Input sanitization blocks malicious payloads.
2. **XSS (Cross-Site Scripting)**: No vulnerable endpoints found.
3. **JWT Tampering**: Invalid tokens correctly rejected.
4. **Password Security:** BCrypt hashing with salt, no plain text storage
5. **Rate Limiting**: High-volume access from a single IP was rate-limited as expected.
6. **Access Control**: Role-based restrictions were strictly enforced by all endpoints.
7. **Data Isolation:** Cross-company data access prevented through service layer validation.

**4.4 Integration Testing**

| **Module** | **Test Case Description** | **Status** |
| --- | --- | --- |
| Frontend – Backend | Real-time expense status updates, authentication flow | Seamless |
| Database Consistency | Transaction integrity during concurrent expense operations | Passed |
| Multi-user Scenario | Simultaneous expense submission and approval | Passed |
| Cross-Browser Support | Web interface tested on Chrome, Firefox, Safari, Edge | Passed |

**4.5 Observations**

1. **Architecture Benefits**: The Spring Boot MVC architecture provided excellent separation of concerns, making the system highly maintainable and testable. Individual components (authentication, expense management, approval workflows) can be modified independently without affecting other modules.
2. **Security Implementation**: JWT-based stateless authentication combined with Spring Security provided robust security without session management complexity. Role-based access control effectively prevents unauthorized operations while maintaining user experience.
3. **Database Design Excellence**: Oracle Database with Hibernate ORM demonstrated excellent performance for complex queries involving multiple table joins. The multi-company relationship model successfully handles enterprise scenarios without data leakage.
4. **User Experience**: The responsive web interface provides an intuitive workflow for both expense submission and approval processes. Real-time status updates and clear navigation enhance user productivity.
5. **Scalability Potential**: The modular design and database optimization support horizontal scaling for larger organizations. Connection pooling and query optimization ensure consistent performance as data volume grows.
6. **Integration Readiness**: RESTful API design with comprehensive error handling makes the system easily integrable with external accounting software, mobile applications, and reporting tools.

****

**CHAPTER 5**

**Conclusion and Future Scope**

**5.1 Conclusion**

The Enterprise Expense Tracker project successfully demonstrates the development of a comprehensive, secure, and scalable expense management solution tailored for modern business needs. It effectively addresses the challenges faced by traditional expense tracking systems—such as manual processing inefficiencies, lack of real-time visibility, security vulnerabilities, and complex multi-company operations—by offering well-structured RESTful APIs and an intuitive web interface for key operations including user authentication, multi-company management, expense submission, approval workflows, and financial reporting.

Through extensive functional, performance, and security testing, the system has proven to be robust under concurrent user loads, secure against common web vulnerabilities, and flexible for integration with accounting software and mobile applications. The Spring Boot-based architecture promotes code reusability, maintainability, and system interoperability, making it suitable for both small startups and large enterprises with complex organizational structures.

**Key Achievements:**

1. **Modern Architecture Implementation**: Successfully implemented enterprise-grade Spring Boot 3.2.0 application with JWT-based authentication, demonstrating current industry best practices.
2. **Multi-Company Support**: Developed sophisticated many-to-many relationships enabling users to operate across multiple companies with different roles, addressing complex business scenarios.
3. **Robust Security Framework**: Implemented comprehensive security measures including JWT authentication, BCrypt password hashing, role-based access control, and protection against common web vulnerabilities.
4. **Database Excellence**: Designed and implemented optimized Oracle Database schema with Flyway migrations, ensuring data integrity and supporting complex business relationships.
5. **User Experience Focus**: Created responsive web interface with intuitive workflows for both expense submission and approval processes, enhancing user productivity and satisfaction.
6. **Scalable Design**: Built modular architecture supporting horizontal scaling and independent component modification, preparing the system for enterprise growth.

**5.2 Future Scope**

While the current system provides a solid foundation for expense management, there is significant potential to extend its capabilities and enhance its value proposition:

**5.2.1 Advanced Integration Capabilities**

1. **Accounting Software Integration**
   * Integrate with popular accounting systems (QuickBooks, SAP, Xero) for automatic expense synchronization
   * Implement real-time financial reporting and budget tracking with accounting software APIs
   * Support for automated journal entry creation and general ledger updates
2. **Mobile Application Development**
   * Develop native iOS and Android applications with offline capability
   * Implement receipt capture using camera functionality with OCR (Optical Character Recognition)
   * Add push notifications for expense approvals and status updates
3. **Cloud Storage Integration**
   * Connect with cloud storage services (Google Drive, OneDrive, Dropbox) for receipt attachments
   * Implement automatic backup and document management capabilities
   * Support for digital receipt storage and retrieval

**5.2.2 Artificial Intelligence and Machine Learning**

1. **AI-Powered Expense Classification**
   * Implement machine learning models for automatic expense category detection based on description and merchant data
   * Smart duplicate expense detection to prevent fraudulent submissions
   * Predictive analytics for budget forecasting and spending pattern analysis
2. **Intelligent Approval Workflows**
   * Machine learning-based risk assessment for automatic approval of low-risk expenses
   * Anomaly detection for unusual spending patterns or potential policy violations
   * Smart routing of expenses to appropriate approvers based on historical patterns
3. **Advanced Reporting and Analytics**
   * AI-driven insights into spending trends and cost optimization opportunities
   * Predictive budget analysis and overspend alerts
   * Automated compliance monitoring and policy violation detection

The implementation of these enhancements would significantly elevate the system's capabilities, making it a comprehensive enterprise expense management platform capable of competing with established solutions while maintaining the flexibility and cost-effectiveness that makes it attractive to growing businesses. The modular architecture established in the current version provides a solid foundation for implementing these advanced features incrementally, ensuring continuous value delivery to users and stakeholders.