PROJECT: EXPENSIFY

IT314 – SOFTWARE ENGINEERING

GROUP 26

TASK: CLASS DIAGRAMS

Devarsh Soni — 202201446

GROUP MEMBERS:

ID	NAME
202201410	SIDDHANT KOTAK
202201419	SHAH MAAHI BIJAL
202201421	KKAVY KAPIL DAVE
202201440	MUNJAPARA DHARMIL JAYANTILAL
202201446	SONI DEVARSH SUSHANT
202201449	LATHIGARA PRIYANSH HARESHBHAI
202201452	DETROJA AVI ASHISHBHAI
202201456	MANAV TIRATH PAREKH
202201458	MAKWANA SRUSHTI NARESHKUMAR

Design Decisions

Sprint 1: User Account Management and Authentication

- **Email Verification**: Emails are validated during registration to prevent fraudulent account creation.
- **Password Recovery**: A time-limited recovery link is sent to users' email for password reset.
- **Language Settings**: Users can dynamically change the interface language using the LocalizationService.

Sprint 2: Core Expense Management Features

- **Expense Tracking:** Expenses are categorized and tracked in real-time for accurate monitoring.
- **Budget Setting:** Users can set flexible daily, weekly, or monthly budgets and track progress.
- **Expenditure Limits:** Daily limits are set by users, and alerts are sent when exceeded.
- **Notifications:** Automated notifications are integrated to alert users about budget overspending.

Sprint 3: Financial Insights and History

- **Expense History:** Users can view a historical list of their expenses, categorized by type and date, for detailed analysis.
- **Expense Reminders:** Users can set recurring reminders for regular expenses to stay on top of payments.
- **Financial Reporting**: Users can generate customizable financial reports based on expense categories, time periods, and other criteria.

Sprint 4: Advanced Financial Management

- **Emergency Funds Management:** Users can set up and track their emergency funds to ensure financial preparedness.
- **Currency Preference:** Users can select their preferred currency, which affects how expenses and budgets are displayed.

- **Dynamic Budget Adjustment:** Budgets automatically adjust based on the user’s spending patterns, allowing for more flexible financial management.

Sprint 5: Invoicing and Payment Features

- **Invoice Generation:** Users can generate invoices with customizable templates for professional and consistent output.
- **Invoice Payment Tracking:** The system tracks payment status (paid, pending, overdue) for each invoice, ensuring up-to-date financial records.
- **Invoice Payment Tracking:** The system tracks payment status (paid, pending, overdue) for each invoice, ensuring up-to-date financial records.

Sprint 6: User and System Administration

- **User Account Management:** Admins have full control over user accounts, including the ability to create, deactivate, or modify accounts.
- **System Updates:** Admins can manage system updates, ensuring that the platform stays secure and functional with the latest features.
- **System Updates:** Admins can manage system updates, ensuring that the platform stays secure and functional with the latest features.
- **System Updates:** Admins can manage system updates, ensuring that the platform stays secure and functional with the latest features.

Sprint 7: Data Security and Enforcement

- **Data Security Enforcement:** Robust encryption and access control policies are implemented to protect sensitive user and system data.
- **Ad Campaign Management:** Users can create, modify, and monitor advertisement campaigns within the system. This ensures efficient management and tracking of marketing efforts.

- **Demographic Targeting:** The system uses demographic data (e.g., age, location, interests) to target advertisements more effectively, optimizing ad relevance for users.

Sprint 8: Advertisement Management and Performance

- **Ad Performance Monitoring:** Users can track key performance metrics (e.g., impressions, clicks, conversions) for ad campaigns, allowing for data-driven decision-making.
- **Analytics Reports:** The system provides detailed reports on ad campaign performance, enabling users to assess the effectiveness of their campaigns and adjust strategies as needed.

Object Oriented Concepts:

1. **Encapsulation**
    - User details like passwordHash and preferredLanguage are encapsulated within the User class. The external system can interact with the user through public methods like register(), login(), and updateLanguage(), but the internal details remain protected.
    - The Expense and Budget classes encapsulate sensitive financial information, such as amount and budgetLimit, allowing controlled access to these details through methods like logExpense() and setBudget().
    - The Expense and Budget classes encapsulate sensitive financial information, such as amount and budgetLimit, allowing controlled access to these details through methods like logExpense() and setBudget().

2. **Abstraction**
    - The AuthenticationService class abstracts the complexity of user authentication, including token generation and validation. Users interact with high-level methods like authenticate() and generateToken(), hiding the details of how the tokens are created.
    - The ExpenseTracker class abstracts the logic of categorizing and tracking expenses. Users interact with simplified

methods like addExpense() and getExpenseHistory(), which abstract away the underlying data structures.

- The ExpenseTracker class abstracts the logic of categorizing and tracking expenses. Users interact with simplified methods like addExpense() and getExpenseHistory(), which abstract away the underlying data structures.

3. **Inheritance**
    - Different types of ads (e.g., DisplayAd, VideoAd) can inherit from a common Ad class, which holds shared attributes like budget, targetAudience, and methods like getAdDetails().
    - Specialized classes like TokenSecurity and EncryptionSecurity inherit from a common SecurityManager base class. Each subclass implements specific methods for handling token management and encryption, maintaining consistency across the system while allowing for specialization.
    - Different types of notifications (e.g., EmailNotification, SMSNotification) inherit from a common Notification class, ensuring that each notification type implements a consistent sendNotification() method.

4. **Polymorphism**
    - Different types of notifications (e.g., EmailNotification, SMSNotification) inherit from a common Notification class, ensuring that each notification type implements a consistent sendNotification() method.
    - Different types of notifications (e.g., EmailNotification, SMSNotification) inherit from a common Notification class, ensuring that each notification type implements a consistent sendNotification() method.
    - Different types of notifications (e.g., EmailNotification, SMSNotification) inherit from a common Notification class, ensuring that each notification type implements a consistent sendNotification() method.