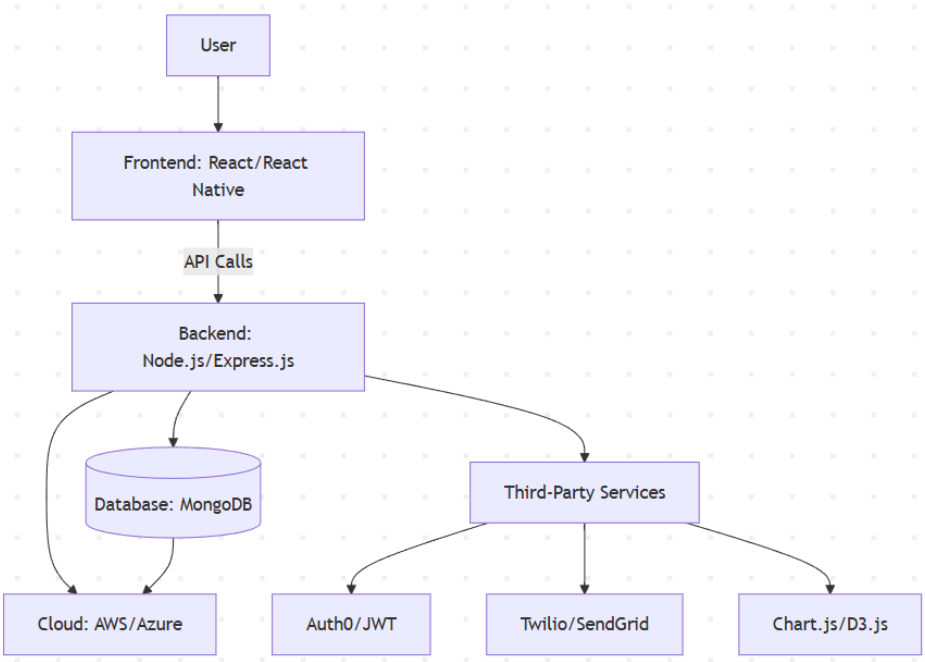
Requirements Gathering and Analysis Phase

Date	15 April 2025
Team ID	SWTID1744094910
Project Title	Personal Expense Tracker App
Maximum Marks	3 Marks

Solution Architecture



Solution Architecture Diagram

1. Frontend Architecture

1.1 Technologies

- Framework: React.js (web) + React Native (mobile) for cross-platform compatibility.
- UI Libraries: Syncfusion Components (grids, charts), Material-UI, Bootstrap.
- State Management: Redux/Context API for global state handling.
- Offline Support: Local storage + Service Workers for offline expense entry.

1.2 Key Modules

- User Authentication: Login, registration, password reset.
- Dashboard: Visual summaries (pie charts, bar graphs) using Chart.js.
- Expense Management: CRUD operations for expenses/categories.
- Multi-User Support: Role-based access control (e.g., Admin, Family Member).

2. Backend Architecture

2.1 Technologies

- Runtime: Node.js with Express.js (RESTful APIs).
- Authentication: JWT tokens with Redis for session management.
- APIs: RESTful endpoints for:
    - `/api/expenses` (CRUD operations)
    - `/api/budgets` (set/update alerts)
    - `/api/reports` (generate PDF/CSV)

2.2 Key Services

- Expense Categorization Engine:
    - Uses Naive Bayes (as per search result3) to auto-category expenses from SMS/bank data.
- Budget Alert System:
    - Real-time notifications via Twilio/SendGrid when thresholds are crossed.
- Report Generation:
    - PDF/Excel exports using libraries like `pdf-lib` or `exceljs`.

3. Database Architecture

3.1 Technologies

- Primary Database: MongoDB (NoSQL) for flexible schema and scalability.
- Schema Design:

3.2 Data Security

- Encryption: AES-256 for data at rest and TLS 1.3 for data in transit.
- Backup: Automated daily backups to AWS S3.

4. Cloud Deployment & DevOps

4.1 Infrastructure

- Frontend: AWS Amplify / Vercel (static hosting).
- Backend: AWS EC2/Azure VM with Docker containers.
- Database: MongoDB Atlas (managed cloud service).

4.2 CI/CD Pipeline

1. GitHub Actions: Automated testing (Jest/Mocha).
2. Docker Hub: Containerization for environment consistency.
3. AWS CodeDeploy: Zero-downtime deployments.

5. Third-Party Integrations

Service	Purpose	Example
Auth0	Secure authentication	Social logins (Google)
Twilio	SMS alerts for budget breaches	SendGrid for emails
Plaid API	Bank transaction sync (future)	Fetch real-time expenses
Chart.js	Data visualization	Pie charts, trend lines

6. Security Architecture

- Authentication: OAuth 2.0 + JWT with refresh tokens.
- Authorization: Role-based access (e.g., Admin vs. Guest).
- Audit Logs: Track user actions for compliance (stored in MongoDB).
- GDPR Compliance: User data deletion/export endpoints.

7. Scalability & Performance

- Horizontal Scaling: Load balancers + auto-scaling groups on AWS.
- Caching: Redis for frequently accessed data (e.g., monthly reports).
- Database Sharding: Split MongoDB collections by region/userGroup.

8. Testing Strategy

Test Type	Tools/Methods	Coverage
Unit Testing	Jest, Mocha	Core logic (expense categorization)
Integration Testing	Postman, Supertest	API endpoints
UAT	User feedback loops	Dashboard usability
Security Testing	OWASP ZAP, Penetration testing	Vulnerability scans