Integration Phase

Date	15 April 2025
Team ID	SWTID1744094910
Project Title	Personal Expense Tracker App
Maximum Marks	2 Marks

API Integrations for Expense Tracker App

1. Plaid API (Bank/Credit Card Integration)

Purpose: Automatically import transactions from banks and credit cards.
Authentication: OAuth 2.0 (via Plaid Link).
Steps:

1. Generate public_key and secret_key from Plaid Dashboard.
2. Initialize Plaid Link in the frontend:

```
// React example
<PlaidLink
  clientName="Expense Tracker"
  env="sandbox"
  product={["transactions"]}
  publicKey="PLAID_PUBLIC_KEY"
  onSuccess={(public_token) => exchangePublicToken(public_token)}
>
  Connect Bank Account
</PlaidLink>
```

3. Backend token exchange:

```
// Node.js example
const exchangePublicToken = async (publicToken) => {
  const response = await plaidClient.exchangePublicToken(publicToken);
  const accessToken = response.data.access_token;
  // Store accessToken securely for future transactions
};
```

4. Fetch transactions:

```
const getTransactions = async (accessToken) => {
  const response = await plaidClient.getTransactions(
    accessToken,
    '2024-01-01',
    '2024-04-30'
  );
  return response.data.transactions;
};
```

Security:

- Store access_token encrypted in the database.
- Use Plaid's webhook to detect account updates.

2. Twilio API (SMS Alerts)

Purpose: Send budget breach alerts via SMS.
Authentication: Twilio SID + Auth Token.
Integration:

```
const sendSMS = (to, message) => {
  const client = require('twilio')('TWILIO_SID', 'TWILIO_AUTH_TOKEN');
  client.messages.create({
    body: message,
    from: '+123456789', // Twilio number
    to: to
  });
};

// Example usage
sendSMS('+919876543210', 'Alert: You've exceeded your Food budget!');
```

Best Practices:

- Rate-limit alerts to avoid spamming users.
- Use templates for consistent messaging.

3. Google Vision API (Receipt OCR)

Purpose: Extract text from receipt images.
Authentication: Google Cloud Service Account Key.
Integration:

```
const { ImageAnnotatorClient } = require('@google-cloud/vision');
const client = new ImageAnnotatorClient({
  keyFilename: 'service-account-key.json'
});

const extractReceiptText = async (imageBuffer) => {
  const [result] = await client.textDetection(imageBuffer);
  return result.textAnnotations[0].description;
};

// Example usage
const receiptText = await extractReceiptText(req.file.buffer);
const parsedData = parseReceipt(receiptText); // Custom logic
```

Cost Optimization:

- Cache results to reduce API calls for duplicate receipts.

4. CurrencyLayer API (Multi-Currency Support)

Purpose: Convert expenses to the user's base currency.
Authentication: API Key.
Integration:

```
const convertCurrency = async (amount, from, to) => {
  const response = await axios.get(
    `http://apilayer.net/api/convert?
access_key=CURRENCYLAYER_KEY&from=${from}&to=${to}&amount=${amount}`
  );
  return response.data.result;
};
```

Caching:

- Store exchange rates daily to avoid hitting API limits.

5. SendGrid API (Email Reports)

Purpose: Send weekly/monthly expense reports via email.
Authentication: API Key.
Integration:

```
const sgMail = require('@sendgrid/mail');
sgMail.setApiKey('SENDGRID_API_KEY');

const sendReport = async (userEmail, reportData) => {
  const msg = {
    to: userEmail,
    from: 'reports@expensetracker.app',
    subject: 'Your Monthly Expense Report',
    html: `<p>Total spent: $${reportData.total}</p>`,
    attachments: [{
      content: reportData.pdfBase64,
      filename: 'report.pdf',
      type: 'application/pdf',
      disposition: 'attachment'
    }]
  };
  await sgMail.send(msg);
};
```

Security & Best Practices

1. Secrets Management:
    - Store API keys in environment variables (e.g., process.env.PLAID_SECRET).
    - Use AWS Secrets Manager or Vault for production.
2. Rate Limiting:
    - Implement retry logic with exponential backoff.
    - Use Redis to track API call quotas.
3. Error Handling:

```
try {
  await convertCurrency(100, 'USD', 'INR');
} catch (error) {
  console.error('Currency API failed:', error.message);
  // Fallback to cached rates
}
```

4. Logging:
    - Log API failures to tools like Datadog or Splunk.

Testing API Integrations

1. Mock APIs: Use Postman or WireMock for offline testing.
2. Unit Tests:

```
// Mock Twilio in Jest
jest.mock('twilio');
test('sends SMS on budget alert', async () => {
  await triggerBudgetAlert();
  expect(twilioClient.messages.create).toHaveBeenCalled();
});
```

3. Monitor Uptime: Use Pingdom or New Relic.

Summary of Integrated Services

API	Use Case	Cost (Monthly)
Plaid	Bank transactions	$500+ (Pro)
Twilio	SMS alerts	$0.0075/SMS
Google Vision	Receipt OCR	$1.50/1000 req
CurrencyLayer	FX conversion	$9.99 (Basic)
SendGrid	Email reports	Free tier (100/day)

This integration plan ensures your app can scale securely while providing real-world value.