FINANALYTICS MINI PROJECT 2

W. Dewi Fitriasari

Institute of Data Programme Assignment

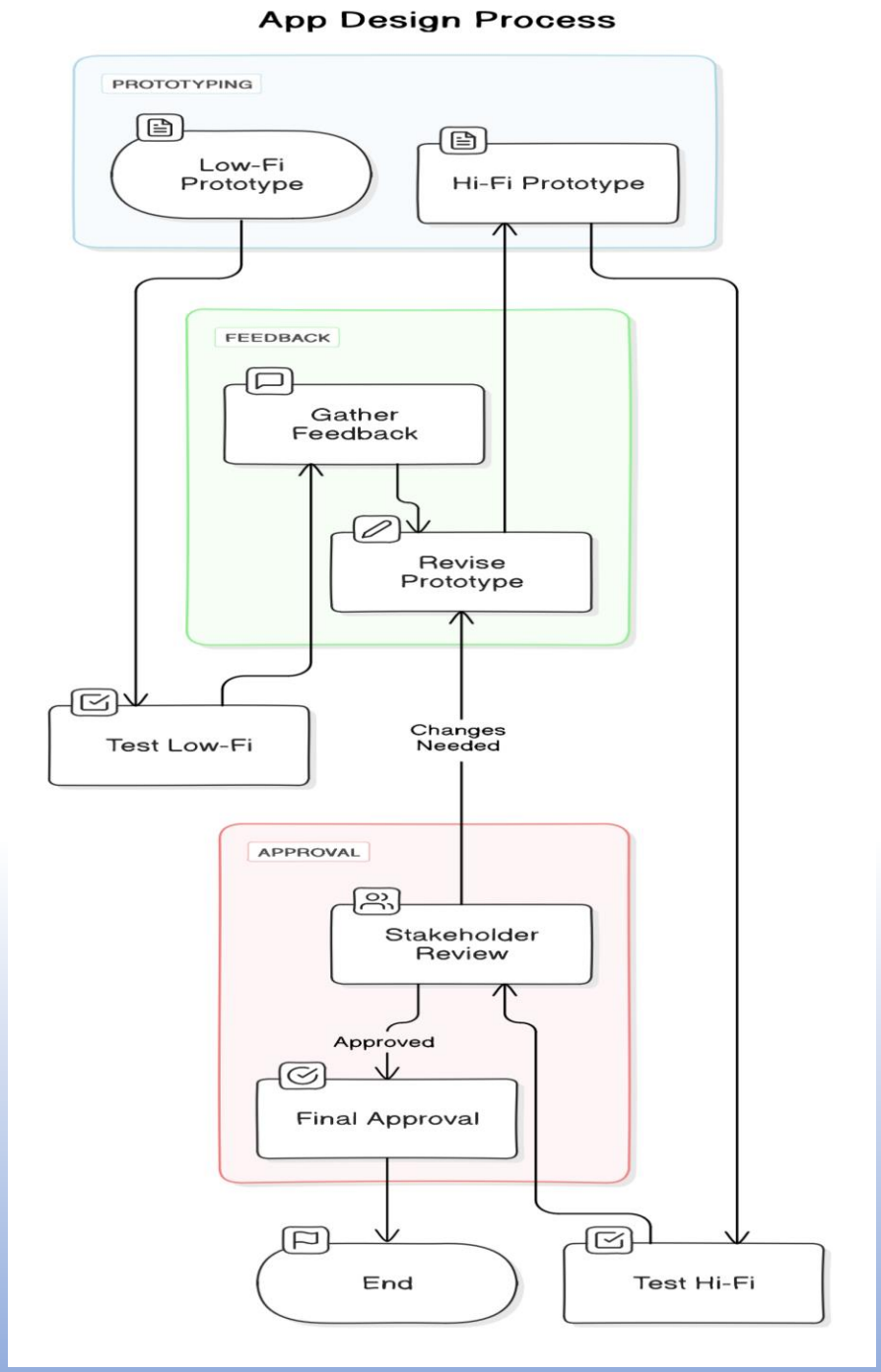
REQUIREMENTS GATHERING AND DESIGN PROCESS

Requirements:

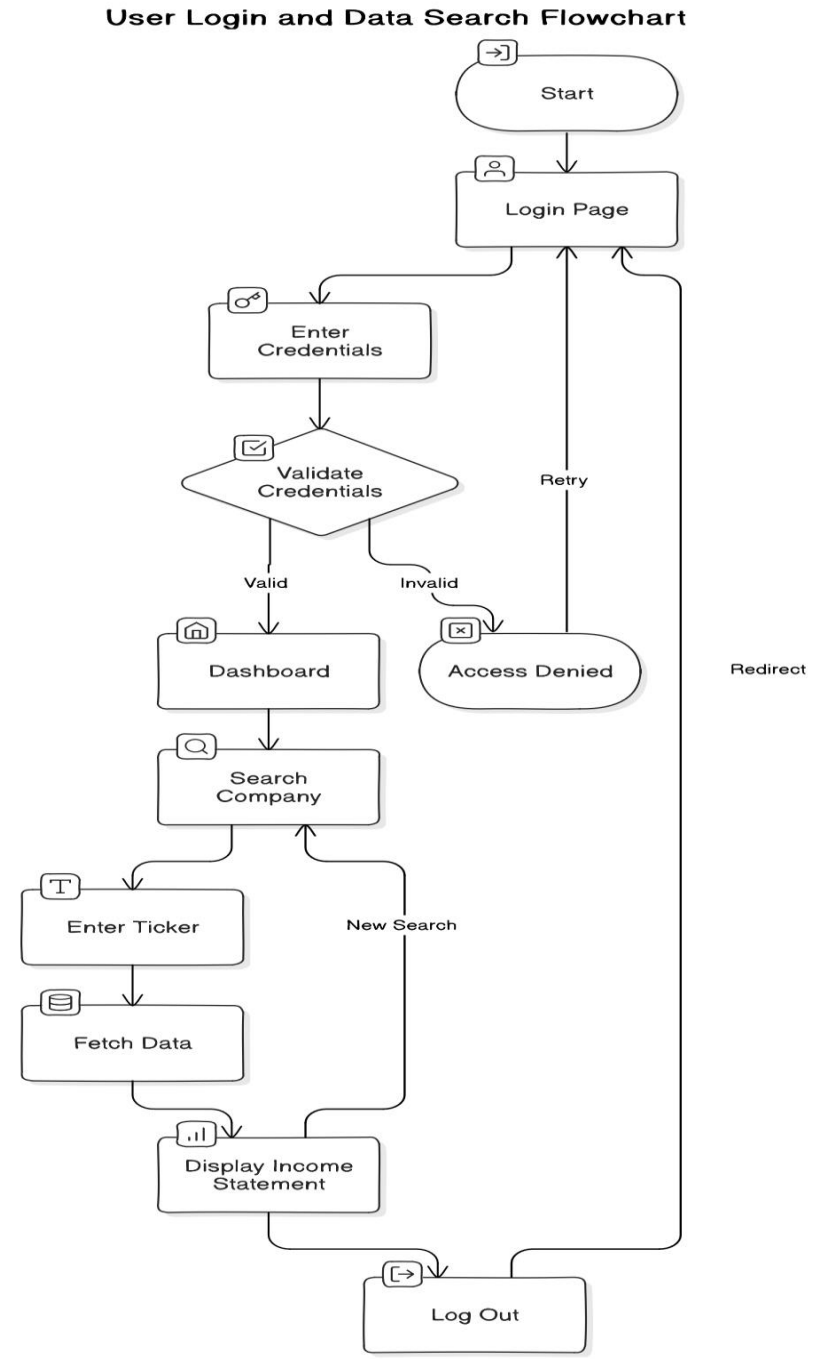
1. Users can search for income statements from NASDAQ-listed companies.
2. All users must have credentials listed in the database of pre-defined users before they can use the search function and retrieve data.

REQUIREMENTS GATHERING AND DESIGN PROCESS

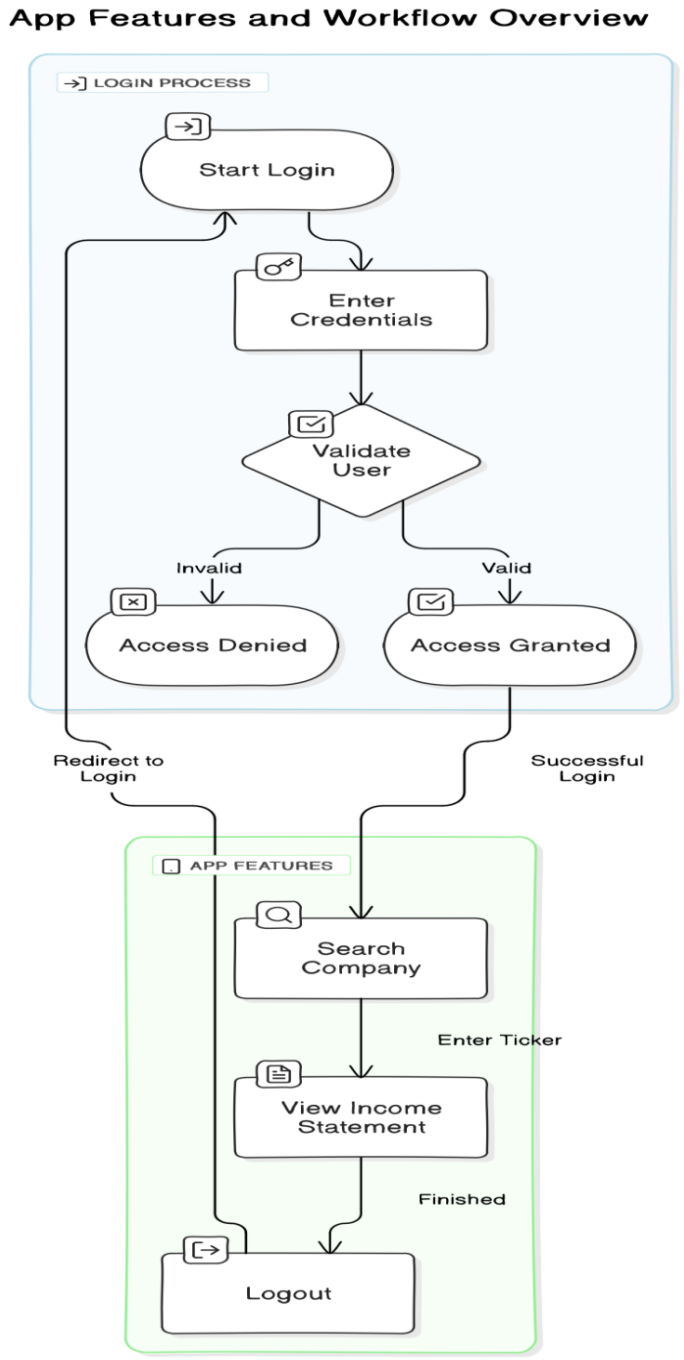
Design Process:



USE CASE


User Login and Data Search Flowchart
Start
Login Page
Enter Credentials
Validate Credentials
Retry
Valid
Invalid
Dashboard
Access Denied
Redirect
Search Company
Enter Ticker
New Search
Fetch Data
Display Income Statement
Log Out

HIGH-LEVEL OVERVIEW OF THE APPLICATION AND ITS FEATURES

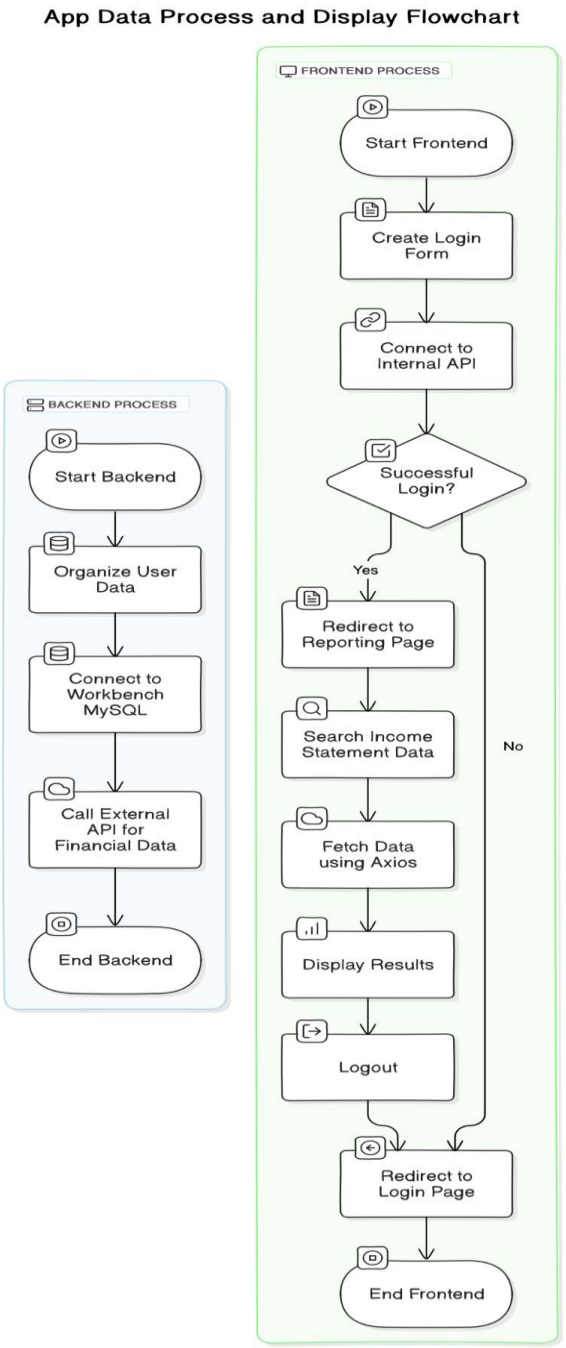


DATA SOURCE

- Pre-defined users: Internal API connected to Workbench MySQL

- Companies' data: External API https://site.financialmodelingprep.com/

DATA PROCESS AND DISPLAY



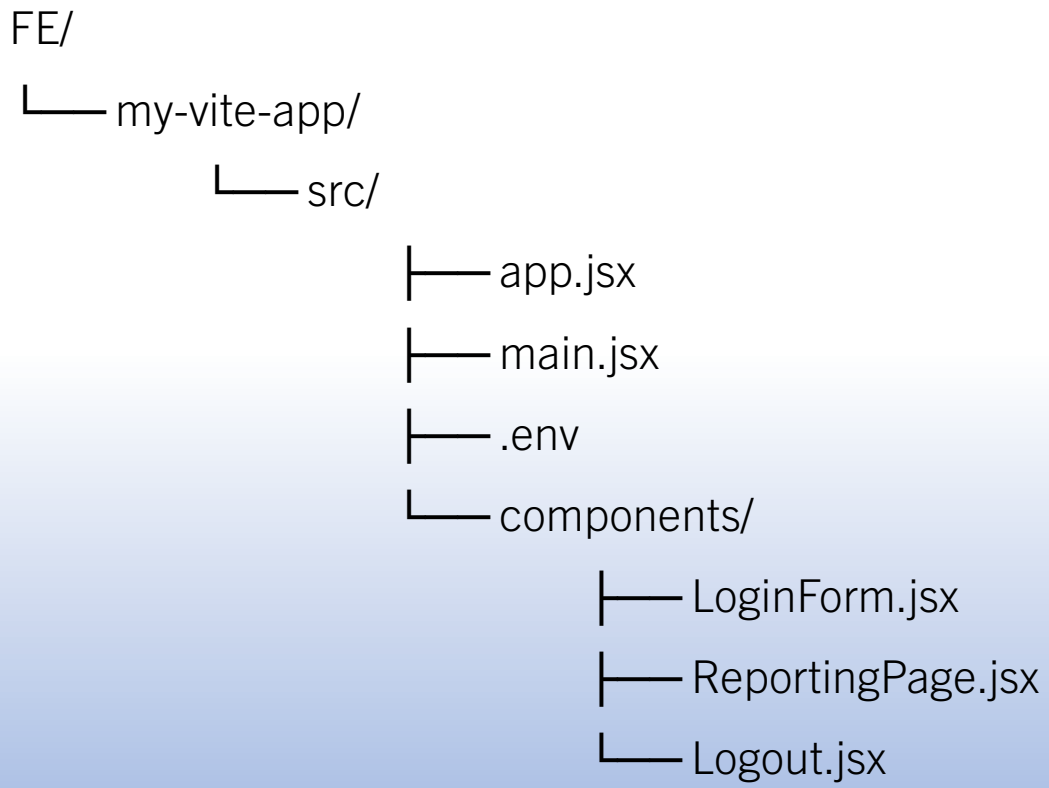
USER INTERACTION WITH THE APP

All users must be registered to maintain their credentials in the API and database. Users can log into the app using their registered email and password.

Users can search for company data by entering the ticker symbol, with AAPL displayed by default.

Once finished, the user clicks the logout button and is re-directed to the login page.

THE STRUCTURE OF THE COMPONENTS/CODES



THE STRUCTURE OF THE COMPONENTS/CODES

```
BE/
└── my-express-api/
        ├── database.js
        ├── server.js
        ├── server.test.js
        └── .babelrc
```

REACT HOOKS USED – LOGINFORM.JSX

useState:
This hook manages the local state for the email and password fields.

Usage:
const [email, setEmail] = useState(""); and
const [password, setPassword] = useState("");

useNavigate:
This hook is part of the React Router and is used to programmatically navigate to a different route after a successful login.

Usage: const navigate = useNavigate();

REACT HOOKS USED – LOGOUT.JSX

useNavigate:
This hook is part of the React Router and allows navigation to a different route when the user logs out.
Usage: const navigate = useNavigate();

REACT HOOKS USED – REPORTINGPAGE.JSX

useState:

This hook is used to manage multiple functions:

- **trendData**: to store the fetched financial data.
- **loading**: to indicate whether data is currently being loaded.
- **error**: to capture any error messages during data fetching.
- **companies**: to maintain the list of company symbols.
- **input**: to handle the input for adding new companies.
- **showOnlyRevenue**: to toggle the display of revenue data only.

REACT HOOKS USED – REPORTINGPAGE.JSX

```
const [trendData, setTrendData] = useState([]);
const [loading, setLoading] = useState(true);
const [error, setError] = useState(null);
const [companies, setCompanies] = useState(["AAPL"]);
const [input, setInput] = useState("");
const [showOnlyRevenue, setShowOnlyRevenue] = useState(false);
```

REACT HOOKS USED – REPORTINGPAGE.JSX

useEffect:
This hook performs side effects to fetch financial data whenever the company's state changes.

Usage:

When the companies state is updated (that is, when a new company symbol is added), the fetchTrendData function is triggered to retrieve the latest total financial data relevant to those companies.

```
useEffect(() => {  fetchTrendData();}, [companies]);
```

TOOLS AND LIBRARIES

REACT

Usage: All components (LoginForm, Logout, ReportingPage, App) are functional components using React.

React Router

Library: react-router-dom

Usage:

Used for routing within the application in the app.jsx

The component Routes, is utilised to define navigation paths.

TOOLS AND LIBRARIES

Material-UI (MUI)
Library: @mui/material
Usage:

Provides pre-built React components for building user interfaces. Components like TextField, Button, Typography, Box, Grid, Paper, Table, CircularProgress, and Alert are used for styling and layout in the UI.

TOOLS AND LIBRARIES

Sequelize
Library: ORM (Object-Relational Mapping) library
Usage:

Used to interact with relational databases (in this case, MySQL).
Used in database.js of the backend

DataTypes (from Sequelize)
Tool: Data types for defining model attributes.
Usage:

Used to specify the data types of the User model attributes (in this case, STRING) for the server.js of the backend

TOOLS AND LIBRARIES

Environment Variables
Tool: process.env
Usage:

Used to access environment variables under database.js of the backend (such as DB_NAME, DB_USER, DB_PASS, and DB_HOST) for database configuration.
This is important for securing sensitive information.

TOOLS AND LIBRARIES

Express
Framework: Web framework for Node.js.
Usage:
Used to create the server and define routes for handling HTTP requests (e.g., app.get, app.post) in the server.js

Axios
Library: AxiosPromise-based HTTP client.
Usage:
Used for making HTTP requests to external APIs, specifically to fetch the financial data from the external API.

TOOLS AND LIBRARIES

Dotenv

Tool: .env, the environment variable management tool feature for node.js

Usage:

Loads environment variables from a .env file into process.env, enabling secure configuration of sensitive data related to the database credentials, API key and port information.

TOOLS AND LIBRARIES

CORS

Type: Middleware for handling Cross-Origin Resource Sharing.

Usage:

Used to allow or restrict resources to be requested from another domain outside the domain from which the first resource was served.
It is in use for the server.js of the backend.
The line app.use(cors()); applies the CORS middleware to the Express application.

TOOLS AND LIBRARIES

Supertest
Library: Testing library for HTTP assertions.
Usage:

Used to make requests to the Express application and assert responses. It simplifies the testing of API endpoints.

Jest
Framework: JavaScript testing framework.
Usage:

The test structure (e.g., describe, it, expect) is using Jest.

FUTURE IMPROVEMENTS OF THE APPLICATION

1. User Authentication & Authorization:

Implement JWT: Use JSON Web Tokens (JWT) for secure authentication and session management.
Role-Based Access Control: Introduce user roles (e.g., admin, user) to manage permissions for different functionalities.

FUTURE IMPROVEMENTS OF THE APPLICATION

2. Data Visualization

Charts & Graphs: Integrate libraries like Chart.js or D3.js to visualize financial data in the ReportingPage.

Dashboards: Create dashboards that provide a summary of financial metrics and trends.

3. Additional API Integrations

More Financial APIs: Integrate with additional financial data sources for more comprehensive data.

Notifications: Implement real-time notifications for important events, such as alerts based on user-defined criteria.

FUTURE IMPROVEMENTS OF THE APPLICATION

Performance Enhancements

Caching: Implement caching to improve data retrieval performance.

Pagination & Lazy Loading: Add pagination for large datasets and lazy loading for performance optimisation.

END OF PRESENTATION

Disclaimer:

This presentation has been made with careful consideration of modules learned from the Software Engineering programme of the Institute of Data. While it is accurate as per assumptions applied and integrated in the application for the mini project, the use of any data and opinions in this document and the related app and APIs, must be with the acknowledgement of the project's developer.