# HABIT TRACKER APPLICATION DOCUMENTATION

## INTRODUCTION

### Introduction of Project

The Habit Tracker Application is a comprehensive web-based system designed to help users track, manage, and analyze their daily habits. The application enables users to create personalized habit tracking routines, monitor their progress over time, and visualize their achievements through interactive reports and statistics. By providing a structured approach to habit formation and maintenance, the application aims to help users develop positive behaviours and achieve their personal goals.

### Existing System and Need for System

Existing habit tracking solutions often lack comprehensive analytics, user-friendly interfaces, or the ability to categorize and customize habits based on individual needs. Many existing applications also fail to provide motivational elements such as badges, streaks, and visual progress indicators that can significantly enhance user engagement and habit adherence.

The need for this system arises from:

- The growing awareness of habit formation as a key factor in personal development and goal achievement

- The lack of comprehensive, user-friendly habit tracking tools that provide detailed analytics

- The need for a system that incorporates behavioral psychology principles to enhance habit formation

- The demand for a responsive, accessible platform that can be used across different devices

### Scope of Work

The Habit Tracker Application includes the following features:

- User registration and authentication system

- Profile management with customizable user information

- Habit creation, categorization, and management

- Daily habit tracking and completion logging

- Progress visualization through interactive charts and reports

- Achievement system with badges and streaks

- Statistical analysis of habit performance

- Responsive design for desktop and mobile devices

### Operating Environment - Hardware and Software

\*\*Hardware Requirements:\*\*

- Server: Any modern server capable of running Node.js applications

- Client: Any device with a modern web browser

- Storage: Minimum 1GB for application and database

- Network: Internet connection for client-server communication

\*\*Software Requirements:\*\*

- Frontend: React.js, Tailwind CSS, Framer Motion

- Backend: Node.js, Express.js

- Database: MongoDB

- Authentication: JWT (JSON Web Tokens)

- Version Control: Git

- Deployment: Any cloud platform supporting Node.js applications

### Detail Description of Technology Used

\*\*Frontend Technologies:\*\*

- \*\*React.js\*\*: A JavaScript library for building user interfaces, providing component-based architecture and efficient DOM manipulation

- \*\*Tailwind CSS\*\*: A utility-first CSS framework for rapidly building custom designs

- \*\*Framer Motion\*\*: A production-ready motion library for React that makes it easy to create animations

- \*\*React Router\*\*: For handling navigation and routing within the application

- \*\*Axios\*\*: For making HTTP requests to the backend API

\*\*Backend Technologies:\*\*

- \*\*Node.js\*\*: A JavaScript runtime built on Chrome's V8 JavaScript engine

- \*\*Express.js\*\*: A minimal and flexible Node.js web application framework

- \*\*MongoDB\*\*: A NoSQL database for storing user and habit data

- \*\*Mongoose\*\*: An Object Data Modeling (ODM) library for MongoDB and Node.js

- \*\*JWT\*\*: For secure authentication and authorization

- \*\*Bcrypt\*\*: For password hashing and security

## PROPOSED SYSTEMS

### Objectives of System

1. To provide users with a simple and intuitive interface for tracking daily habits

2. To enable categorization and customization of habits based on individual needs

3. To visualize progress and achievements through interactive charts and reports

4. To motivate users through gamification elements such as badges and streaks

5. To provide statistical insights into habit formation and adherence patterns

6. To ensure data security and user privacy through robust authentication

7. To deliver a responsive experience across different devices and screen sizes

### Proposed System

The proposed Habit Tracker Application is a full-stack web application with a React.js frontend and Node.js/Express backend. The system follows a client-server architecture where:

1. \*\*Client-Side (Frontend):\*\*

- Provides user interface for all interactions

- Handles form validation and data presentation

- Manages client-side state and routing

- Renders visualizations and reports

- Communicates with the server via RESTful API calls

2. \*\*Server-Side (Backend):\*\*

- Handles authentication and authorization

- Processes and validates incoming requests

- Interacts with the database for CRUD operations

- Implements business logic for habit tracking and analysis

- Returns appropriate responses to the client

3. \*\*Database:\*\*

- Stores user information, habits, and tracking data

- Maintains relationships between different data entities

- Ensures data integrity and consistency

### User Requirements

1. \*\*Authentication and User Management:\*\*

- User registration with email verification

- Secure login with password protection

- Password reset functionality

- Profile management and customization

2. \*\*Habit Management:\*\*

- Create, read, update, and delete habits

- Categorize habits (health, productivity, learning, etc.)

- Set frequency (daily, weekly) and time of day

- Add descriptions and notes to habits

3. \*\*Habit Tracking:\*\*

- Mark habits as complete

- View current and historical completion status

- Track streaks and consistency

- Receive badges for achievements

4. \*\*Reporting and Analytics:\*\*

- View weekly and monthly completion rates

- Analyze performance by category

- Track progress over time

- Generate visual reports of habit data

5. \*\*User Interface:\*\*

- Responsive design for all devices

- Intuitive navigation and interaction

- Visual feedback for user actions

- Accessibility compliance

## ANALYSIS AND DESIGN

### Entity Relationship Diagram (ERD)

The ERD for the Habit Tracker Application consists of the following main entities:

1. \*\*User\*\*

2. \*\*Habit\*\*

3. \*\*Completion\*\*

4. \*\*Badge\*\*

5. \*\*Category\*\*

The relationships between these entities are:

- A User can have many Habits (one-to-many)

- A Habit belongs to one User (many-to-one)

- A Habit can have many Completions (one-to-many)

- A Habit can have many Badges (many-to-many)

- A Habit belongs to one Category (many-to-one)

```plantuml

@startuml

entity "User" as user {

\*id : ObjectId

--

\*name : String

\*email : String

\*password : String

profilePicture : String

isVerified : Boolean

createdAt : Date

updatedAt : Date

}

entity "Habit" as habit {

\*id : ObjectId

--

\*userId : ObjectId

\*title : String

description : String

\*category : String

\*frequency : String

timeOfDay : String

currentStreak : Number

longestStreak : Number

lastCompleted : Date

createdAt : Date

updatedAt : Date

}

entity "Completion" as completion {

\*id : ObjectId

--

\*habitId : ObjectId

\*userId : ObjectId

\*date : Date

createdAt : Date

}

entity "Badge" as badge {

\*id : ObjectId

--

\*habitId : ObjectId

\*name : String

\*icon : String

description : String

createdAt : Date

}

entity "Category" as category {

\*id : String

--

\*name : String

icon : String

}

user ||--o{ habit : has

habit ||--o{ completion : has

habit ||--o{ badge : earns

category ||--o{ habit : contains

@enduml

```

### Data Flow Diagram (DFD)

#### Level 0 DFD (Context Diagram)

```plantuml

@startuml

!define ENTITY entity

!define PROCESS process

ENTITY "User" as user

PROCESS "Habit Tracker System" as system

user --> system : User Registration/Login

user --> system : Habit Management

user --> system : Habit Tracking

system --> user : Authentication Response

system --> user : Habit Data

system --> user : Reports & Analytics

@enduml

```

#### Level 1 DFD

```plantuml

@startuml

!define ENTITY entity

!define PROCESS process

!define DATASTORE database

ENTITY "User" as user

PROCESS "Authentication System" as auth

PROCESS "Habit Management" as habitMgmt

PROCESS "Habit Tracking" as habitTrack

PROCESS "Reporting System" as report

DATASTORE "User Database" as userDB

DATASTORE "Habit Database" as habitDB

DATASTORE "Completion Database" as completionDB

user --> auth : Login/Register

auth --> user : Auth Response

auth --> userDB : Store/Retrieve User Data

user --> habitMgmt : Create/Update/Delete Habits

habitMgmt --> habitDB : Store/Retrieve Habit Data

habitMgmt --> user : Habit Data

user --> habitTrack : Mark Habit Complete

habitTrack --> completionDB : Store Completion Data

habitTrack --> habitDB : Update Streak Data

habitTrack --> user : Tracking Confirmation

user --> report : Request Reports

report --> habitDB : Retrieve Habit Data

report --> completionDB : Retrieve Completion Data

report --> user : Visual Reports

@enduml

```

### Object Diagram

```plantuml

@startuml

object User {

id = "60d21b4967d0d8992e610c85"

name = "John Doe"

email = "john@example.com"

profilePicture = "/uploads/profiles/john.jpg"

isVerified = true

}

object Habit1 {

id = "60d21b4967d0d8992e610c86"

userId = "60d21b4967d0d8992e610c85"

title = "Morning Meditation"

description = "15 minutes of mindfulness meditation"

category = "mindfulness"

frequency = "daily"

timeOfDay = "morning"

currentStreak = 5

longestStreak = 10

}

object Habit2 {

id = "60d21b4967d0d8992e610c87"

userId = "60d21b4967d0d8992e610c85"

title = "Read a Book"

description = "Read for 30 minutes"

category = "learning"

frequency = "daily"

timeOfDay = "evening"

currentStreak = 3

longestStreak = 7

}

object Completion1 {

id = "60d21b4967d0d8992e610c88"

habitId = "60d21b4967d0d8992e610c86"

userId = "60d21b4967d0d8992e610c85"

date = "2023-06-15"

}

object Completion2 {

id = "60d21b4967d0d8992e610c89"

habitId = "60d21b4967d0d8992e610c86"

userId = "60d21b4967d0d8992e610c85"

date = "2023-06-16"

}

object Badge1 {

id = "60d21b4967d0d8992e610c90"

habitId = "60d21b4967d0d8992e610c86"

name = "5-Day Streak"

icon = "🔥"

description = "Completed habit for 5 consecutive days"

}

User -- Habit1

User -- Habit2

Habit1 -- Completion1

Habit1 -- Completion2

Habit1 -- Badge1

@enduml

```

### Class Diagram

```plantuml

@startuml

class User {

-id: ObjectId

-name: String

-email: String

-password: String

-profilePicture: String

-isVerified: Boolean

-createdAt: Date

-updatedAt: Date

+register(): void

+login(): Token

+updateProfile(): void

+resetPassword(): void

+verifyEmail(): void

}

class Habit {

-id: ObjectId

-userId: ObjectId

-title: String

-description: String

-category: String

-frequency: String

-timeOfDay: String

-currentStreak: Number

-longestStreak: Number

-lastCompleted: Date

-badges: Badge[]

-createdAt: Date

-updatedAt: Date

+create(): void

+update(): void

+delete(): void

+complete(): void

+calculateStreak(): void

+earnBadge(): void

}

class Completion {

-id: ObjectId

-habitId: ObjectId

-userId: ObjectId

-date: Date

-createdAt: Date

+create(): void

+delete(): void

}

class Badge {

-id: ObjectId

-habitId: ObjectId

-name: String

-icon: String

-description: String

-createdAt: Date

+create(): void

}

class Category {

-id: String

-name: String

-icon: String

+getAll(): Category[]

}

class Report {

+generateWeeklyReport(userId: ObjectId): Report

+generateMonthlyReport(userId: ObjectId): Report

+generateCategoryReport(userId: ObjectId): Report

+calculateCompletionRate(habits: Habit[]): Number

}

User "1" -- "many" Habit : has >

Habit "1" -- "many" Completion : has >

Habit "1" -- "many" Badge : earns >

Category "1" -- "many" Habit : contains >

Report -- Habit : analyzes >

Report -- Completion : analyzes >

@enduml

```

### Use Case Diagrams

```plantuml

@startuml

left to right direction

actor "User" as user

rectangle "Habit Tracker System" {

usecase "Register" as UC1

usecase "Login" as UC2

usecase "Manage Profile" as UC3

usecase "Create Habit" as UC4

usecase "Update Habit" as UC5

usecase "Delete Habit" as UC6

usecase "Mark Habit Complete" as UC7

usecase "View Progress" as UC8

usecase "Generate Reports" as UC9

usecase "View Badges" as UC10

usecase "Filter Habits" as UC11

usecase "Sort Habits" as UC12

}

user --> UC1

user --> UC2

user --> UC3

user --> UC4

user --> UC5

user --> UC6

user --> UC7

user --> UC8

user --> UC9

user --> UC10

user --> UC11

user --> UC12

UC7 ..> UC10 : <<include>>

UC8 ..> UC11 : <<extend>>

UC8 ..> UC12 : <<extend>>

UC9 ..> UC8 : <<include>>

@enduml

```

### Activity Diagram

```plantuml

@startuml

start

:User opens application;

if (User is logged in?) then (yes)

:Display Dashboard;

else (no)

:Display Login Screen;

:User enters credentials;

if (Credentials valid?) then (yes)

:Authenticate User;

:Display Dashboard;

else (no)

:Show Error Message;

stop

endif

endif

:User navigates application;

fork

:Create New Habit;

:Fill Habit Details;

:Save Habit;

fork again

:View Existing Habits;

if (Mark habit as complete?) then (yes)

:Update Habit Status;

:Update Streak;

if (Achievement earned?) then (yes)

:Award Badge;

else (no)

endif

else (no)

endif

fork again

:View Reports;

:Select Report Type;

fork

:View Weekly Report;

fork again

:View Monthly Report;

fork again

:View Category Report;

end fork

end fork

:User logs out;

stop

@enduml

```

### Collaboration Diagram

```plantuml

@startuml

actor User

User -> AuthController : login(email, password)

AuthController -> UserModel : findByEmail(email)

UserModel --> AuthController : user

AuthController -> AuthController : validatePassword(password)

AuthController --> User : authToken

User -> HabitController : createHabit(habitData)

HabitController -> HabitModel : create(habitData)

HabitModel --> HabitController : newHabit

HabitController --> User : habitDetails

User -> HabitController : completeHabit(habitId)

HabitController -> CompletionModel : create(habitId, userId, date)

CompletionModel --> HabitController : completion

HabitController -> HabitModel : updateStreak(habitId)

HabitModel --> HabitController : updatedHabit

HabitController -> BadgeModel : checkAndCreateBadges(habit)

BadgeModel --> HabitController : newBadges

HabitController --> User : completionStatus, newBadges

User -> ReportController : generateReport(userId, type)

ReportController -> HabitModel : findByUserId(userId)

HabitModel --> ReportController : habits

ReportController -> CompletionModel : findByUserId(userId)

CompletionModel --> ReportController : completions

ReportController -> ReportController : processData(habits, completions, type)

ReportController --> User : reportData

@enduml

```

### Deployment Diagram

```plantuml

@startuml

node "Client Device" {

[Web Browser] as browser

[React Application] as reactApp

}

node "Web Server" {

[Node.js] as nodejs

[Express.js] as express

[API Endpoints] as api

}

node "Database Server" {

[MongoDB] as mongodb

}

node "File Storage" {

[User Uploads] as uploads

}

browser -- reactApp : HTTPS

reactApp -- api : REST API/HTTPS

api -- nodejs : runs on

nodejs -- express : uses

express -- mongodb : Mongoose ODM

nodejs -- uploads : File I/O

@enduml

```

### Component Diagram

```plantuml

@startuml

package "Frontend" {

[Authentication Components] as authComp

[Habit Management Components] as habitComp

[Tracking Components] as trackComp

[Reporting Components] as reportComp

[UI Components] as uiComp

[State Management] as stateComp

[API Service] as apiService

}

package "Backend" {

[Authentication Controllers] as authController

[Habit Controllers] as habitController

[Tracking Controllers] as trackController

[Reporting Controllers] as reportController

[Middleware] as middleware

[Database Models] as models

[Utility Services] as utils

}

database "MongoDB" {

[Users Collection] as users

[Habits Collection] as habits

[Completions Collection] as completions

[Badges Collection] as badges

}

authComp --> apiService

habitComp --> apiService

trackComp --> apiService

reportComp --> apiService

uiComp --> stateComp

apiService --> authController : HTTP

apiService --> habitController : HTTP

apiService --> trackController : HTTP

apiService --> reportController : HTTP

authController --> middleware

habitController --> middleware

trackController --> middleware

reportController --> middleware

middleware --> models

models --> users

models --> habits

models --> completions

models --> badges

authController --> utils

habitController --> utils

trackController --> utils

reportController --> utils

@enduml

```

### Table Design

#### User Table

| Field | Type | Constraints | Description |

|-------|------|-------------|-------------|

| \_id | ObjectId | Primary Key | Unique identifier for the user |

| name | String | Required | User's full name |

| email | String | Required, Unique | User's email address |

| password | String | Required | Hashed password |

| profilePicture | String | Optional | Path to profile picture |

| isVerified | Boolean | Default: false | Email verification status |

| createdAt | Date | Auto | Record creation timestamp |

| updatedAt | Date | Auto | Record update timestamp |

#### Habit Table

| Field | Type | Constraints | Description |

|-------|------|-------------|-------------|

| \_id | ObjectId | Primary Key | Unique identifier for the habit |

| userId | ObjectId | Foreign Key, Required | Reference to User |

| title | String | Required | Habit title |

| description | String | Optional | Habit description |

| category | String | Required | Habit category |

| frequency | String | Required | Daily, Weekly, etc. |

| timeOfDay | String | Optional | Morning, Afternoon, Evening, etc. |

| currentStreak | Number | Default: 0 | Current consecutive completions |

| longestStreak | Number | Default: 0 | Longest streak achieved |

| lastCompleted | Date | Optional | Date of last completion |

| createdAt | Date | Auto | Record creation timestamp |

| updatedAt | Date | Auto | Record update timestamp |

#### Completion Table

| Field | Type | Constraints | Description |

|-------|------|-------------|-------------|

| \_id | ObjectId | Primary Key | Unique identifier for completion |

| habitId | ObjectId | Foreign Key, Required | Reference to Habit |

| userId | ObjectId | Foreign Key, Required | Reference to User |

| date | Date | Required | Date of completion |

| createdAt | Date | Auto | Record creation timestamp |

#### Badge Table

| Field | Type | Constraints | Description |

|-------|------|-------------|-------------|

| \_id | ObjectId | Primary Key | Unique identifier for badge |

| habitId | ObjectId | Foreign Key, Required | Reference to Habit |

| name | String | Required | Badge name |

| icon | String | Required | Badge icon (emoji or path) |

| description | String | Optional | Badge description |

| createdAt | Date | Auto | Record creation timestamp |

### Data Dictionary

#### User Entity

- \*\*\_id\*\*: Unique identifier for the user (MongoDB ObjectId)

- \*\*name\*\*: User's full name (String, Required)

- \*\*email\*\*: User's email address, used for login (String, Required, Unique)

- \*\*password\*\*: Hashed password for security (String, Required)

- \*\*profilePicture\*\*: Path to the user's profile picture (String, Optional)

- \*\*isVerified\*\*: Indicates if the user's email is verified (Boolean, Default: false)

- \*\*createdAt\*\*: Timestamp when the user account was created (Date, Auto-generated)

- \*\*updatedAt\*\*: Timestamp when the user account was last updated (Date, Auto-generated)

#### Habit Entity

- \*\*\_id\*\*: Unique identifier for the habit (MongoDB ObjectId)

- \*\*userId\*\*: Reference to the user who owns the habit (ObjectId, Required)

- \*\*title\*\*: Name of the habit (String, Required)

- \*\*description\*\*: Detailed description of the habit (String, Optional)

- \*\*category\*\*: Category the habit belongs to (String, Required)

- \*\*frequency\*\*: How often the habit should be performed (String, Required)

- \*\*timeOfDay\*\*: When during the day the habit should be performed (String, Optional)

- \*\*currentStreak\*\*: Number of consecutive days the habit has been completed (Number, Default: 0)

- \*\*longestStreak\*\*: Longest streak achieved for this habit (Number, Default: 0)

- \*\*lastCompleted\*\*: Date when the habit was last marked as complete (Date, Optional)

- \*\*createdAt\*\*: Timestamp when the habit was created (Date, Auto-generated)

- \*\*updatedAt\*\*: Timestamp when the habit was last updated (Date, Auto-generated)

#### Completion Entity

- \*\*\_id\*\*: Unique identifier for the completion record (MongoDB ObjectId)

- \*\*habitId\*\*: Reference to the habit that was completed (ObjectId, Required)

- \*\*userId\*\*: Reference to the user who completed the habit (ObjectId, Required)

- \*\*date\*\*: Date when the habit was completed (Date, Required)

- \*\*createdAt\*\*: Timestamp when the completion record was created (Date, Auto-generated)

#### Badge Entity

- \*\*\_id\*\*: Unique identifier for the badge (MongoDB ObjectId)

- \*\*habitId\*\*: Reference to the habit the badge is associated with (ObjectId, Required)

- \*\*name\*\*: Name of the badge (String, Required)

- \*\*icon\*\*: Visual representation of the badge (String, Required)

- \*\*description\*\*: Explanation of how the badge was earned (String, Optional)

- \*\*createdAt\*\*: Timestamp when the badge was awarded (Date, Auto-generated)

### Drawbacks and Limitations

1. \*\*Offline Functionality\*\*: The current system requires an internet connection to function. There is no offline mode for tracking habits when connectivity is unavailable.

2. \*\*Mobile App Absence\*\*: While the web application is responsive, there is no dedicated native mobile application, which might limit the user experience on mobile devices.

3. \*\*Limited Social Features\*\*: The application lacks social features such as sharing achievements or competing with friends, which could enhance motivation and engagement.

4. \*\*Notification System\*\*: The current implementation has limited notification capabilities, which could reduce user engagement and habit adherence.

5. \*\*Data Visualization Complexity\*\*: Some of the more advanced data visualization features may be computationally intensive and could affect performance on lower-end devices.

6. \*\*Limited Integration\*\*: The system does not integrate with other health or productivity applications, limiting its utility within a broader ecosystem of tools.

7. \*\*Scalability Concerns\*\*: As the user base grows, the current architecture may face challenges in maintaining performance and responsiveness.

### Proposed Enhancements

1. \*\*Offline Mode\*\*: Implement local storage and synchronization capabilities to allow users to track habits without an internet connection.

2. \*\*Native Mobile Applications\*\*: Develop dedicated iOS and Android applications to provide a more seamless mobile experience.

3. \*\*Social Features\*\*: Add functionality for users to connect with friends, share achievements, and participate in challenges.

4. \*\*Advanced Notification System\*\*: Implement a comprehensive notification system with customizable reminders and motivational messages.

5. \*\*AI-Powered Insights\*\*: Integrate machine learning algorithms to provide personalized insights and recommendations based on user behavior patterns.

6. \*\*Third-Party Integrations\*\*: Develop APIs and integrations with popular health, fitness, and productivity applications.

7. \*\*Gamification Expansion\*\*: Enhance the gamification elements with more badges, levels, and rewards to increase engagement.

8. \*\*Data Export and Import\*\*: Add functionality for users to export their data or import data from other habit tracking applications.

9. \*\*Advanced Analytics\*\*: Implement more sophisticated analytics tools for deeper insights into habit formation and adherence patterns.

10. \*\*Accessibility Improvements\*\*: Enhance accessibility features to ensure the application is usable by people with various disabilities.

### Conclusion

The Habit Tracker Application provides a comprehensive solution for users to track, manage, and analyze their habits. By combining an intuitive user interface with powerful tracking and reporting capabilities, the system helps users develop and maintain positive behaviors.

The application's key strengths include its user-friendly design, comprehensive habit categorization, visual progress tracking, and motivational elements such as badges and streaks. These features work together to create an engaging experience that encourages consistent habit formation.

While there are some limitations, such as the lack of offline functionality and native mobile applications, the proposed enhancements outline a clear path for future development. By addressing these limitations and implementing the suggested improvements, the Habit Tracker Application can continue to evolve and better serve its users' needs.

The system's architecture, built on modern web technologies like React.js, Node.js, and MongoDB, provides a solid foundation for scalability and future expansion. The comprehensive documentation, including detailed diagrams and data models, ensures that the system can be maintained and enhanced by development teams.

In conclusion, the Habit Tracker Application represents a valuable tool for individuals seeking to improve their lives through consistent habit formation, offering both immediate utility and significant potential for future growth and enhancement.

### Bibliography

1. Clear, J. (2018). Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones. Avery.

2. Duhigg, C. (2012). The Power of Habit: Why We Do What We Do in Life and Business. Random House.

3. Fogg, B.J. (2020). Tiny Habits: The Small Changes That Change Everything. Houghton Mifflin Harcourt.

4. React Documentation. (2023). Retrieved from https://reactjs.org/docs/getting-started.html

5. Node.js Documentation. (2023). Retrieved from https://nodejs.org/en/docs/

6. MongoDB Documentation. (2023). Retrieved from https://docs.mongodb.com/

7. Express.js Documentation. (2023). Retrieved from https://expressjs.com/

8. Tailwind CSS Documentation. (2023). Retrieved from https://tailwindcss.com/docs

9. JWT.io. (2023). Introduction to JSON Web Tokens. Retrieved from https://jwt.io/introduction/

10. Mongoose Documentation. (2023). Retrieved from https://mongoosejs.com/docs/

## ANNEXURES

### ANNEXURE 1: USER INTERFACE SCREENS

#### Login Screen

The login screen provides a secure entry point to the application, featuring email and password fields, a "Forgot Password" link, and an option to register for new users.

#### Registration Screen

The registration screen allows new users to create an account by providing their name, email, and password, with appropriate validation for each field.

#### Dashboard

The dashboard displays the user's habits organized by category, with options to filter and sort. Each habit card shows the title, description, category, current streak, and a button to mark it as complete.

#### Habit Creation/Edit Form

This form allows users to create new habits or edit existing ones, with fields for title, description, category, frequency, and time of day.

#### Progress Page

The progress page displays various charts and statistics about the user's habit completion rates, streaks, and overall performance.

#### Reports Page

The reports page offers different types of reports (weekly, monthly, category) with interactive visualizations of the user's habit data.

#### Profile Page

The profile page allows users to view and edit their personal information, including name, email, and profile picture.

### ANNEXURE 2: OUTPUT REPORTS WITH DATA

#### Weekly Habit Completion Report

This report displays a bar chart showing the percentage of habits completed for each day of the current week, allowing users to identify patterns in their habit adherence.

#### Monthly Habit Completion Report

This report presents a calendar view of the current month, with color-coding to indicate the number of habits completed each day, providing a visual representation of consistency.

#### Category Distribution Report

This report shows how habits are distributed across different categories and the completion rates for each category, helping users identify which areas they are most successful in.

#### Streak Analysis Report

This report displays information about current and longest streaks for each habit, motivating users to maintain their consistency.

#### Badge Collection Report

This report showcases all the badges earned by the user, organized by habit, serving as a visual representation of achievements and milestones.