**Technical Design Document**

**Abstract**

Buddy saving is a system that incorporates savings among friends, partners, colleagues and family. With an account created, you can create a savings plan that allows you to invite multiple members, ranging up to 5 different people, giving you a simple yet conducive saving system.

**Architecture**

The API will be built using a NodeJS server with Typescript as the language of choice. It will use the Express framework for handling requests and responses and Awilix for dependency injection. The API will be designed to be RESTful, with endpoints for creating user accounts, logging in, creating group savings plans, sending invites, and joining group savings plans.

**User Authentication and Authorization**

User authentication and authorization will be implemented using JSON Web Tokens (JWTs) and Bcrypt for password encryption. When a user creates an account or logs in, a JWT will be issued to the client, which will be used to authenticate the user for all subsequent requests. The JWT will be included in the Authorization header of each request.

**Data Storage**

Data will be stored in a PostgreSQL database using Sequelize as the ORM and the packages pg and pg-hstore. The following tables will be created:

* Users: stores user data, including email, password (hashed), and JWT secret.
* group\_savings\_plans: stores group savings plan data, including the user ID of the creator, name, description, goal amount, and current amount.
* Invites: stores invitation data, including the group savings plan ID, inviter ID, invitee ID, and status.

**API Endpoints**

The following API endpoints will be implemented:

* POST /users: Create a new user account. The request body is validated using Joi.
* POST /users/login: logs in a user and issues a JWT. The request body is validated using Joi.
* GET /users/username/:username: Get a user data by username.
* POST /savings: creates a new group savings plan.
* GET /savings/: retrieves a group savings plans.
* GET /savings/:id: retrieves a group savings plan by ID.
* POST /invite: Send invites up to 5 friends to join a group savings plan.
* PUT /invite: Respond to join a group savings plan invite.
* GET: /invite/:id/:inviteId: View invite to join a group savings plan invite.
* GET /invite/sent: retrieves sent invites.
* GET /invite/received: retrieves sent invites

**Error Handling**

Errors will be handled by returning appropriate HTTP status codes and error messages in the response body, with HTTP-status-code to ensure proper HTTP status for responses. The following error codes will be used:

* 400 Bad Request: the request was malformed or contained invalid data. Joi is used for validating the request body, query, and params.
* 401 Unauthorized: the client was not authorized to access the resource.
* 404 Not Found: the requested resource was not found.
* 409 Conflict: the request could not be completed because it conflicted with an existing resource.
* 429 Too Many Requests: the rate limit has been reached. Express-rate-limit limits the rate of API calls to a reasonable amount per minute, hour or day.
* 500 Internal Server Error: an unexpected error occurred on the server.

**Security Considerations**

The following security considerations will be taken into account:

* Passwords will be hashed using a secure hashing algorithm (e.g., bcrypt).
* JWTs will be signed using a secret unique to each user and stored in the database.
* User input will be validated and sanitized to prevent injection attacks. Joi is used for validating the request body, query, and params.
* The API will be hosted over HTTPS to prevent eavesdropping and man-in-the-middle attacks.
* CORS is used to control access.
* Express-rate-limit is used to limit the rate of API calls to a reasonable amount per minute,